SN8PC20

用户参考手册

Version 1.2

SONiX 8 位单片机

SONiX 公司保留对以下所有产品在可靠性，功能和设计方面的改进作进一步说明的权利。SONiX 不承担由本手册所涉及的产品或电路的运用和使用所引起的任何责任，SONiX 的产品不是专门设计来应用于外科植入、生命维持和任何 SONiX 产品的故障会对个体造成伤害甚至死亡的领域。如果将 SONiX 的产品应用于上述领域，即使这些是由 SONiX 在产品设计和制造上的疏忽引起的，用户应赔偿所有费用、损失、合理的人身伤害或死亡所直接或间接产生的律师费用，并且用户保证 SONiX 及其雇员、子公司、分支机构和销售商与上述事宜无关。

修改记录

版本	日期	修改说明
VER 1.0	2008 年 3 月	初版。
	2008 年 6 月	修改烧录信息章节内容。
VER 1.1	2009 年 5 月	修改了部分章节内容。 增加 IHRC 频率曲线。
VER 1.2	2010 年 6 月	增加完毕低速时钟晶体 32KHz 模式。

目 录

1 产品简介

1.1 功能特性

- **存储器配置**
  OTP ROM 空间：2K * 16 位。
  RAM 空间：56 字节。

- **4 层堆栈缓存器**

- **I/O 配置**
  输入输出双向端口：P0、P1、P5。
  400mA IR 输出引脚：P5.4/IROUT。
  具有唤醒功能的端口：P0、P1 的电平变化触发。
  内置上拉电阻的端口：P0、P1、P5。
  外部中断引脚：P0.0，由 PEDGE 控制。

- **1 级 LVD**

- **Fcpu（指令周期）**
  Fcpu = Fosc/1，Fpsc/2，Fosc/4，Fosc/8。

- **强大的指令系统**
  单周期指令系统（1T）
  绝大部分指令只需要一个周期。
  JMP 指令可寻址整个 ROM 区。
  CALL 指令可寻址整个 ROM 区。
  查表指令 MOVC 可寻址整个 ROM 区。

- **2 个中断源**
  1 个内部中断：T0。
  1 个外部中断：INT0。

- **1 个 8 位定时器**
  T0：基本定时器。

- **单通道红外波形信号输出**

- **5 种系统时钟**
  外部高速时钟：RC 模式，高达 8 MHz。
  外部高速时钟：晶体模式，高达 8 MHz。
  外部低速时钟：晶振模式，32KHz 和 455KHz。
  内部高速时钟：RC 模式，高达 8MHz。
  内部低速时钟：RC 模式，10KHz（3V）。

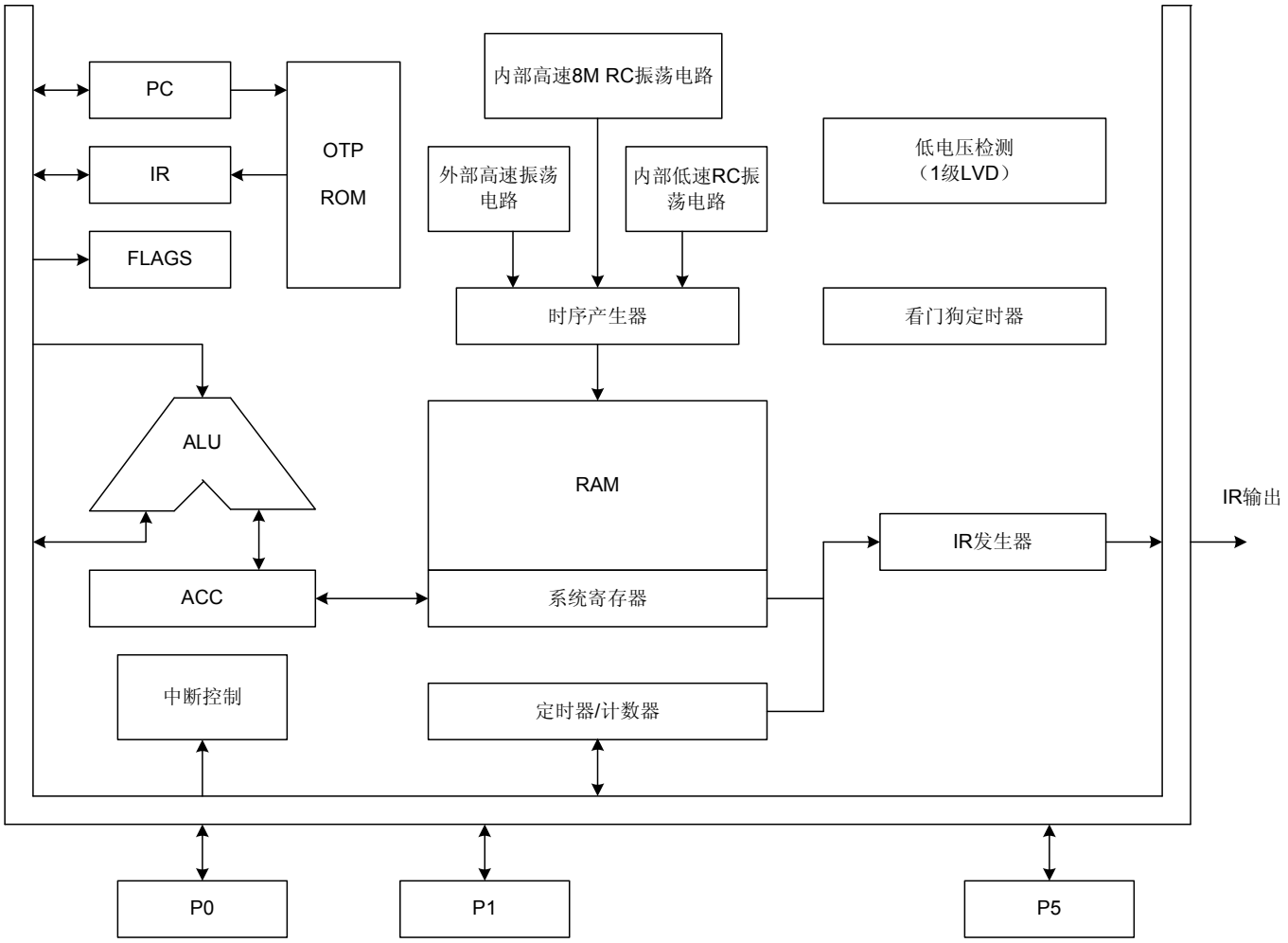
- **工作模式**
  普通模式：高、低速时钟同时工作。
  低速模式：只有低速时钟在工作。
  睡眠模式：高、低速时钟同时停止工作。
  绿色模式：由定时器周期性的唤醒。

- **封装形式**
  PDIP 20 pin。
  SOP 20 pin。
  SSOP 20 pin。

☞ **特性比较表**

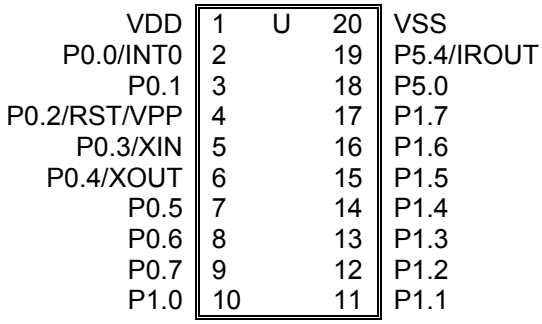
单片机型号	ROM	RAM	堆栈	T0	振荡器			I/O	IR 输出	唤醒功能引脚数目	封装形式
					Ext. 455K	Ext. 4M	Int. 8M				
SN8PC01	0.5K*16	32	4	-	V	-	-	16	固定为 38KHz，正常的电流	9	PDIP20/SOP20
SN8PC13	2K*16	48	4	V	-	V	-	16	频率、占空比可编程控制，正常的电流	15	PDIP20/SOP20/SSOP20
SN8PC20	2K*16	56	4	V	V	V	V	18	频率、占空比可编程控制，400mA 的灌电流	16	PDIP20/SOP20/SSOP20

1.2 系统框图


PC
IR
FLAGS
OTP
ROM
内部高速8M RC振荡电路
外部高速振荡电路
内部低速RC振荡电路
低电压检测
（1级LVD）
时序产生器
看门狗定时器
ALU
RAM
IR输出
IR发生器
ACC
系统寄存器
中断控制
定时器/计数器
P0
P1
P5

1.3 引脚配置

SN8PC20P (PDIP 20 pins)
SN8PC20S (SOP 20 pins)
SN8PC20X (SSOP 20 pins)

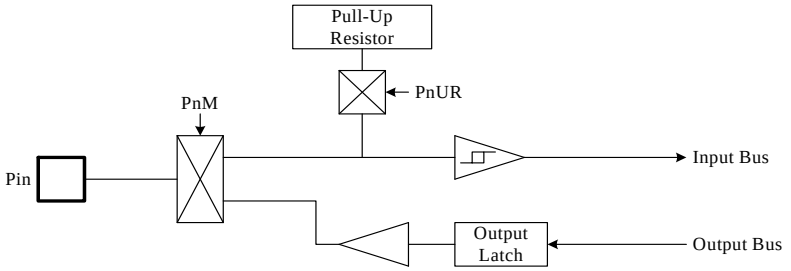


1.4 引脚说明

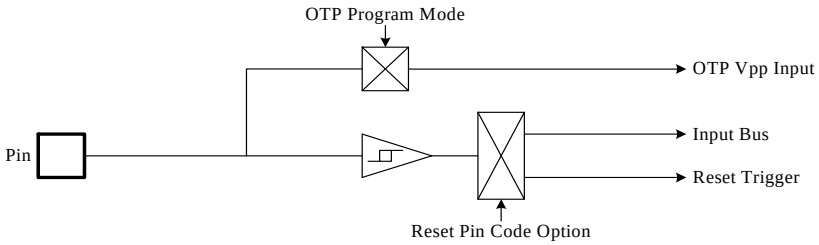
引脚名称	类型	功能说明
VDD, VSS	P	电源输入端。
P0.2/RST/VPP	I, P	RST：系统外部复位输入引脚，施密特触发，低电平有效，通常保持高电平。
		VPP：编程模式下为 OTP 电源输入引脚。
		P0.2：单向输入引脚，施密特触发，无上拉电阻。
XIN/P0.3	I/O	XIN：振荡信号输入引脚。
		P0.3：双向输入输出引脚，施密特触发，内置上拉电阻，输入模式时具有唤醒功能。
XOUT/P0.4	I/O	XOUT：振荡信号输出引脚。
		P0.4：双向输入输出引脚，施密特触发，内置上拉电阻，输入模式时具有唤醒功能。
P0.0/INT0	I/O	P0.0：双向输入输出引脚，施密特触发，内置上拉电阻，输入模式时具有唤醒功能。
		INT0：外部中断触发引脚。
P0.1, P0[7:5]	I/O	P0：双向输入输出引脚，施密特触发，内置上拉电阻，输入模式时具有唤醒功能。
P1[7:0]	I/O	双向输入输出引脚，施密特触发，内置上拉电阻，输入模式时具有唤醒功能，由寄存器 P1W 控制其唤醒功能。
P5.0	I/O	双向输入输出引脚，施密特触发，输入模式时内置上拉电阻。
P5.4/IROUT	I/O	P5.4：双向输入输出引脚，施密特触发，输入模式时内置上拉电阻。
		IROUT：IR 信号输出引脚，频率、占空比可编程控制。

1.5 引脚电路结构图

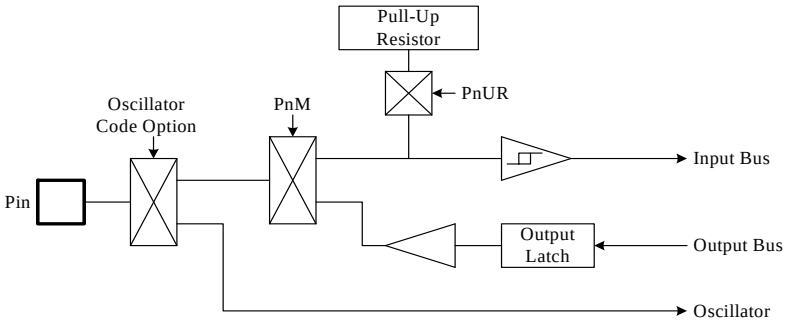
- **I/O 引脚，施密特触发（ViH=0.7*Vdd，ViL=0.3*Vdd），内置上拉电阻（200KΩ@3V）：**



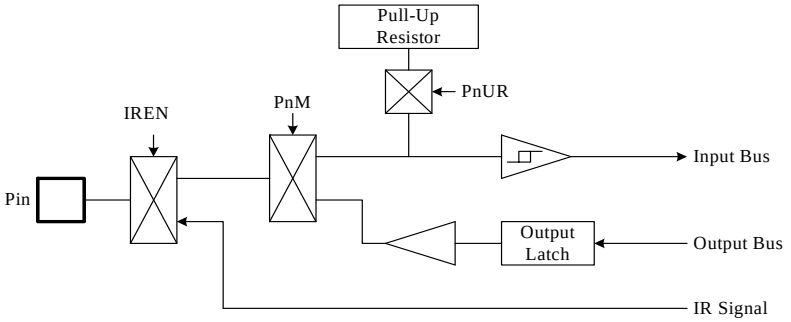
- **单向输入引脚，施密特触发（ViH=0.9*Vdd，ViL=0.2*Vdd），Vpp 高电压：**



- **与振荡器引脚共用的 I/O 引脚，施密特触发（ViH=0.7*Vdd，ViL=0.3*Vdd），内置上拉电阻（200KΩ@3V）：**



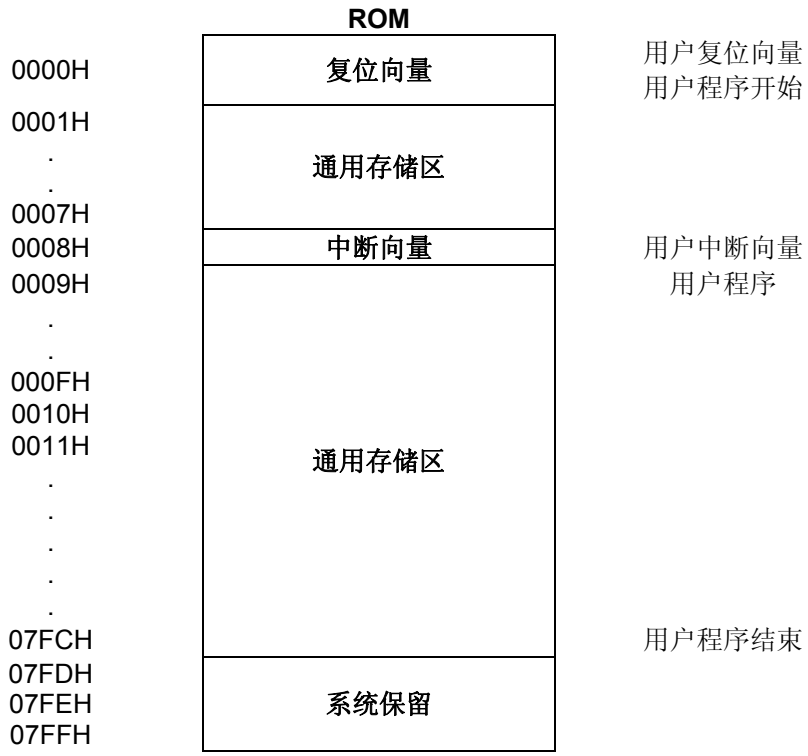
- **与 IR 输出引脚共用的 I/O 引脚，施密特触发（ViH=0.7*Vdd，ViL=0.3*Vdd），内置上拉电阻（200KΩ@3V）：**



2 中央处理器（CPU）

2.1 程序存储器（ROM）

☞ **ROM：2K**



ROM 包括复位向量、中断向量、通用存储区域和系统保留部分。复位向量位于程序的开始处，中断向量则位于中断程序的开始处，通用存储区域则存放用户程序。

2.1.1 复位向量（0000H）

具有一个字长的系统复位向量（0000H）。

- **上电复位（NT0=1，NPD=0）；**
- **看门狗复位（NT0=0，NPD=0）；**
- **外部复位（NT0=1，NPD=1）。**

发生上述任一种复位后，程序将从 0000H 处重新开始执行，系统寄存器也都将恢复为默认值。根据 PFLAG 寄存器中的 NT0 和 NPD 标志位的内容可以判断系统复位方式。下面一段程序演示了如何定义 ROM 中的复位向量。

➢ **例：定义复位向量。**

```
        ORG     0               ;
        JMP     START           ; 跳至用户程序。
        …

        ORG     10H
START:                          ; 用户程序起始地址。
        …                       ; 用户程序。
        …
        ENDP                    ; 程序结束。
```

2.1.2 中断向量（0008H）

中断向量地址为 0008H。一旦有中断响应，程序计数器 PC 的当前值就会存入堆栈缓存器并跳转到 0008H 开始执行中断服务程序。0008H 处的第一条指令必须是“JMP”或“NOP”。下面的示例程序说明了如何编写中断服务程序。

✷ **注：“PUSH”，“POP”指令用于存储和恢复 ACC/PFLAG，NT0、NTD 不受影响。PUSH/POP 缓存器是唯一的，且仅有一层。**

➢ **例：定义中断向量，中断服务程序紧随 ORG 8H 之后。**

```
.CODE
                ORG         0
                JMP         START           ; 跳至用户程序。
                …
                ORG         8H              ; 中断向量。
                PUSH                        ; 保存 ACC 和 PFLAG。
                …
                POP                         ; 恢复 ACC 和 PFLAG。
                RETI                        ; 中断结束。
                …
START:                                      ; 用户程序开始。
                …                           ;
                JMP         START           ; 用户程序结束。
                …
                ENDP                        ; 程序结束。
```

➢ **例：定义中断向量，中断程序在用户程序之后。**

```
.CODE
                ORG         0
                JMP         START           ; 跳至用户程序。
                …
                ORG         8H              ; 中断向量。
                JMP         MY_IRQ          ; 跳至中断程序。

                ORG         10H
START:                                      ; 用户程序开始。
                …
                JMP         START           ; 用户程序结束。
                …
MY_IRQ:                                     ; 中断程序开始。
                PUSH                        ; 保存 ACC 和 PFLAG。
                …
                POP                         ; 恢复 ACC 和 PFLAG。
                RETI                        ; 中断程序结束。
                …
                ENDP                        ; 程序结束。
```

✷ **注：从上面的程序中容易得出 SONiX 的编程规则，有以下几点：**
1、地址 0000H 的“JMP”指令使程序从头开始执行；
2、地址 0008H 是中断向量；
3、用户的程序应该是一个循环。

2.1.3 查表

在 SONiX 单片机中，对 ROM 区中的数据进行查找，寄存器 Y 指向所找数据地址的中间字节（bit8~bit15），寄存器 Z 指向所找数据地址的低字节（bit0~bit7）。执行完 MOVC 指令后，所查找数据低字节内容被存入 ACC 中，而数据高字节内容被存入 R 寄存器。

➢ **例：查找 ROM 地址为“TABLE1”的值。**

```
                B0MOV     Y, #TABLE1$M    ; 设置 TABLE1 地址高字节。
                B0MOV     Z, #TABLE1$L    ; 设置 TABLE1 地址低字节。
                MOVC                      ; 查表，R = 00H，ACC = 35H。

                                          ; 查找下一地址。
                INCMS     Z
                JMP       @F              ; Z 没有溢出。
                INCMS     Y               ; Z 溢出（FFH → 00），→ Y=Y+1
                NOP                       ;
                                          ;
@@:             MOVC                      ; 查表，R = 51H，ACC = 05H。
                …                         ;
TABLE1:         DW        0035H           ; 定义数据表（16 位）数据。
                DW        5105H
                DW        2012H
                …
```

✷ **注：当寄存器 Z 溢出（从 0FFH 变为 00H）时，寄存器 Y 并不会自动加 1。因此， Z 溢出时，Y 必须由程序加 1，下面的宏 INC_YZ 能够对 Y 和 Z 寄存器自动处理。**

➢ **例：宏 INC_YZ。**

```
INC_YZ          MACRO
                INCMS     Z
                JMP       @F              ; 没有溢出。

                INCMS     Y
                NOP                       ; 没有溢出。
@@:
                ENDM
```

➢ **例：通过“INC_YZ”对上例进行优化。**

```
                B0MOV     Y, #TABLE1$M    ; 设置 TABLE1 地址中间字节。
                B0MOV     Z, #TABLE1$L    ; 设置 TABLE1 地址低字节。
                MOVC                      ; 查表，R = 00H，ACC = 35H。

                 INC_YZ                   ; 查找下一地址数据。
                                          ;
@@:             MOVC                      ; 查表，R = 51H，ACC = 05H。
                …                         ;
TABLE1:         DW        0035H           ; 定义数据表（16 位）数据。
                DW        5105H
                DW        2012H
                …
```

下面的程序通过累加器对 Y，Z 寄存器进行处理来实现查表功能，但需要特别注意进位时的处理。

➢ **例：由指令 B0ADD/ADD 对 Y 和 Z 寄存器加 1。**

```
                B0MOV   Y, #TABLE1$M    ; 设置 TABLE1 地址中间字节。
                B0MOV   Z, #TABLE1$L    ; 设置 TABLE1 地址低字节。

                B0MOV   A, BUF          ; Z = Z + BUF。
                B0ADD   Z, A

                B0BTS1  FC              ; 检查进位标志。
                JMP     GETDATA         ; FC = 0。
                INCMS   Y               ; FC = 1。
                NOP
GETDATA:                                ;
                MOVC                    ; 存储数据，如果 BUF = 0，数据为 0035H。
                                        ; 如果 BUF = 1，数据=5105H。
                                        ; 如果 BUF = 2，数据=2012H。
                …
TABLE1:         DW      0035H           ; 定义数据表（16 位）数据。
                DW      5105H
                DW      2012H
                …
```

2.1.4 跳转表

跳转表能够实现多地址跳转功能。由于 PCL 和 ACC 的值相加即可得到新的 PCL，因此，可以通过对 PCL 加上不同的 ACC 值来实现多地址跳转。ACC 值若为 n，PCL+ACC 即表示当前地址加 n，执行完当前指令后 PCL 值还会自加 1，可参考以下范例。如果 PCL＋ACC 后发生溢出，PCH 则自动加 1。由此得到的新的 PC 值再指向跳转指令列表中新的地址。这样，用户就可以通过修改 ACC 的值轻松实现多地址的跳转。

✷ **注：PCH 只支持 PC 增量运算，而不支持 PC 减量运算。当 PCL+ACC 后如有进位，PCH 的值会自动加 1。PCL-ACC 后若有借位，PCH 的值将保持不变，用户在设计应用时要加以注意。**

➢ **例：跳转表。**

```
        ORG     0100H           ; 跳转表从 ROM 前端开始。

        B0ADD   PCL, A          ; PCL = PCL + ACC，PCL 溢出时 PCH 加 1。
        JMP     A0POINT         ; ACC = 0，跳至 A0POINT。
        JMP     A1POINT         ; ACC = 1，跳至 A1POINT。
        JMP     A2POINT         ; ACC = 2，跳至 A2POINT。
        JMP     A3POINT         ; ACC = 3，跳至 A3POINT。
```

SONiX 单片机提供一个宏以保证可靠执行跳转表功能，它会自动检测 ROM 边界并将跳转表移至适当的位置。但采用该宏程序会占用部分 ROM 空间。

➢ **例：如果跳转表跨越 ROM 边界，将引起程序错误。**

```
@JMP_A  MACRO   VAL
        IF      (($+1) !& 0XFF00) !!= (($+(VAL)) !& 0XFF00)
        JMP     ($ | 0XFF)
        ORG     ($ | 0XFF)
        ENDIF
        B0ADD   PCL, A
        ENDM
```

✷ **注：“VAL”为跳转表列表中列表个数。**

➢ **例：宏“MACRO3.H”中，“@JMP_A”的应用。**

```
        B0MOV   A, BUF0         ; “BUF0”从 0 至 4。
        @JMP_A  5               ; 列表个数为 5。
        JMP     A0POINT         ; ACC = 0，跳至 A0POINT。
        JMP     A1POINT         ; ACC = 1，跳至 A1POINT。
        JMP     A2POINT         ; ACC = 2，跳至 A2POINT。
        JMP     A3POINT         ; ACC = 3，跳至 A3POINT。
        JMP     A4POINT         ; ACC = 4，跳至 A4POINT。
```

如果跳转表恰好位于 ROM BANK 边界处（00FFH~0100H），宏指令“@JMP_A”将调整跳转表到适当的位置（0100H）。

➢ **例：“@JMP_A”运用举例。**

```
; 编译前
ROM 地址
                B0MOV     A, BUF0       ; “BUF0”从 0 到 4。
                @JMP_A    5             ; 列表个数为 5。
00FDH           JMP       A0POINT       ; ACC = 0，跳至 A0POINT。
00FEH           JMP       A1POINT       ; ACC = 1，跳至 A1POINT。
00FFH           JMP       A2POINT       ; ACC = 2，跳至 A2POINT。
0100H           JMP       A3POINT       ; ACC = 3，跳至 A3POINT。
0101H           JMP       A4POINT       ; ACC = 4，跳至 A4POINT。

; 编译后
ROM 地址
                B0MOV     A, BUF0       ; “BUF0”从 0 到 4。
                @JMP_A    5             ; 列表个数为 5。
0100H           JMP       A0POINT       ; ACC = 0，跳至 A0POINT。
0101H           JMP       A1POINT       ; ACC = 1，跳至 A1POINT。
0102H           JMP       A2POINT       ; ACC = 2，跳至 A2POINT。
0103H           JMP       A3POINT       ; ACC = 3，跳至 A3POINT。
0104H           JMP       A4POINT       ; ACC = 4，跳至 A4POINT。
```

2.1.5 CHECKSUM计算

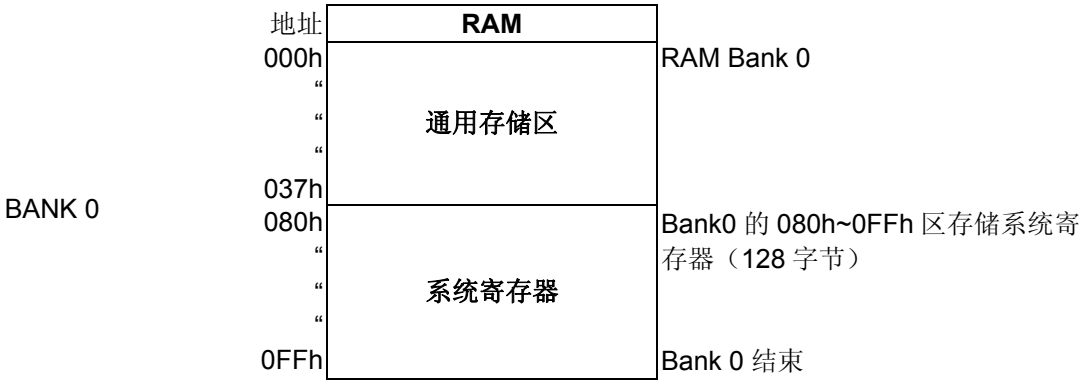
ROM 区末端位置的几个字限制使用，进行 Checksum 计算时，用户应避免对该单元的访问。

➢ **例：示例程序演示了如何对 00H 到用户程序结束进行 Checksum 计算。**

```
                MOV     A,#END_USER_CODE$L
                B0MOV   END_ADDR1, A        ; 用户程序结束地址低位地址存入 end_addr1。
                MOV     A,#END_USER_CODE$M
                B0MOV   END_ADDR2, A        ; 用户程序结束地址中间地址存入 end_addr2。
                CLR     Y                   ; 清 Y。
                CLR     Z                   ; 清 Z。
@@:
                MOVC
                B0BCLR  FC                  ; 清标志位 C。
                ADD     DATA1, A            ;
                MOV     A, R
                ADC     DATA2, A            ;
                JMP     END_CHECK           ; 检查 YZ 地址是否为代码的结束地址。
AAA:
                INCMS   Z
                JMP     @B                  ; 若 Z != 00H，进行下一个计算。
                JMP     Y_ADD_1             ; 若 Z = 00H，Y+1。
END_CHECK:
                MOV     A, END_ADDR1
                CMPRS   A, Z                ; 检查 Z 地址是否为用户程序结束地址低位地址。
                JMP     AAA                 ; 否，则进行 Checksum 计算。
                MOV     A, END_ADDR2
                CMPRS   A, Y                ; 是则检查 Y 的地址是否为用户程序结束地址中间地址。
                JMP     AAA                 ; 否，则进行 Checksum 计算。
                JMP     CHECKSUM_END        ; 是则 Checksum 计算结束。

Y_ADD_1:
                INCMS   Y                   ;
                NOP
                JMP     @B                  ; 跳转到 Checksum 计算。
CHECKSUM_END:
                …
                …
```

2.2 数据存储器（RAM）

☞ **RAM：56 字节**



Sonix 提供“Bank0”类型的指令（如：B0MOV、B0ADD、B0BTS1、B0BSET 等）来直接访问 Bank0。

2.2.1 系统寄存器

2.2.1.1 系统寄存器列表

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
8	L	H	R	Z	Y	-	PFLAG	-	-	-	-	-	-	-	-	-
9	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
A	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
B	-	-	-	-	-	-	-	-	P0M	-	-	-	-	-	-	PEDGE
C	P1W	P1M	-	-	-	P5M	-	-	INTRQ	INTEN	OSCM	-	WDTR	IRR	PCL	PCH
D	P0	P1	-	-	-	P5	-	-	T0M	T0C	IRM	IRC	-	-	-	STKP
E	P0UR	P1UR	-	-	-	P5UR	@HL	@YZ	IRD	-	-	-	-	-	-	-
F	-	-	-	-	-	-	-	-	STK3L	STK3H	STK2L	STK2H	STK1L	STK1H	STK0L	STK0H

2.2.1.2 系统寄存器说明

H, L = 专用寄存器，@HL 间接寻址寄存器，ROM 寻址寄存器
R = 工作寄存器和 ROM 查表数据缓存器
PnM = Pn 输入/输出模式控制寄存器
INTRQ = 中断请求寄存器
OSCM = 振荡模式控制寄存器
IRR = IR 自动装载数据缓存器
Pn = Pn 数据缓存器
T0C = T0 计数寄存器
IRC = IR 计数寄存器
PnUR = Pn 上拉电阻控制寄存器
@YZ = 间接寻址寄存器
STK0~STK3 = 堆栈缓存器
Y, Z = 专用寄存器，@YZ 间接寻址寄存器，ROM 寻址寄存器
PFLAG = ROM 页及特殊标志寄存器
PEDGE = P0.0 触发模式控制寄存器
INTEN = 中断使能寄存器
WDTR = 看门狗定时器清零寄存器
PCH, PCL = 程序计数器
T0M = T0 模式寄存器
IRM = IR 模式寄存器
STKP = 堆栈指针寄存器
@HL = 间接寻址寄存器
IRD = IR 占空比控制寄存器

2.2.1.3 系统寄存器的位定义

地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	R/W	注释
080H	LBIT7	LBIT6	LBIT5	LBIT4	LBIT3	LBIT2	LBIT1	LBIT0	R/W	L
081H	HBIT7	HBIT6	HBIT5	HBIT4	HBIT3	HBIT2	HBIT1	HBIT0	R/W	H
082H	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0	R/W	R
083H	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0	R/W	Z
084H	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0	R/W	Y
086H	NT0	NPD				C	DC	Z	R/W	PFLAG
0B8H	P07M	P06M	P05M	P04M	P03M		P01M	P00M	R/W	P0M
0BFH				P00G1	P00G0				R/W	PEDGE
0C0H	P17W	P16W	P15W	P14W	P13W	P12W	P11W	P10W	W	P1W
0C1H	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M	R/W	P1M
0C5H				P54M				P50M	R/W	P5M
0C8H				T0IRQ				P00IRQ	R/W	INTRQ
0C9H				T0IEN				P00IEN	R/W	INTEN
0CAH				CPUM1	CPUM0	CLKMD	STPHX		R/W	OSCM
0CCH	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0	W	WDTR
0CDH	IRR7	IRR6	IRR5	IRR4	IRR3	IRR2	IRR1	IRR0	W	IRR
0CEH	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	R/W	PCL
0CFH						PC10	PC9	PC8	R/W	PCH
0D0H	P07	P06	P05	P04	P03	P02	P01	P00	R/W	P0
0D1H	P17	P16	P15	P14	P13	P12	P11	P10	R/W	P1
0D5H				P54				P50	R/W	P5
0D8H	T0ENB	T0rate2	T0rate1	T0rate0					R/W	T0M
0D9H	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0	R/W	T0C
0DAH							IREN	CREN	R/W	IRM
0DBH	IRC7	IRC6	IRC5	IRC4	IRC3	IRC2	IRC1	IRC0	R/W	IRC
0DFH	GIE						STKPB1	STKPB0	R/W	STKP
0E0H	P07R	P06R	P05R	P04R	P03R		P01R	P00R	W	P0UR
0E1H	P17R	P16R	P15R	P14R	P13R	P12R	P11R	P10R	W	P1UR
0E5H				P54R				P50R	W	P5UR
0E6H	@HL7	@HL6	@HL5	@HL4	@HL3	@HL2	@HL1	@HL0	R/W	@HL
0E7H	@YZ7	@YZ6	@YZ5	@YZ4	@YZ3	@YZ2	@YZ1	@YZ0	R/W	@YZ
0E8H	IRD7	IRD6	IRD5	IRD4	IRD3	IRD2	IRD1	IRD0	W	IRD
0F8H	S3PC7	S3PC6	S3PC5	S3PC4	S3PC3	S3PC2	S3PC1	S3PC0	R/W	STK3L
0F9H						S3PC10	S3PC9	S3PC8	R/W	STK3H
0FAH	S2PC7	S2PC6	S2PC5	S2PC4	S2PC3	S2PC2	S2PC1	S2PC0	R/W	STK2L
0FBH						S2PC10	S2PC9	S2PC8	R/W	STK2H
0FCH	S1PC7	S1PC6	S1PC5	S1PC4	S1PC3	S1PC2	S1PC1	S1PC0	R/W	STK1L
0FDH						S1PC10	S1PC9	S1PC8	R/W	STK1H
0FEH	S0PC7	S0PC6	S0PC5	S0PC4	S0PC3	S0PC2	S0PC1	S0PC0	R/W	STK0L
0FFH						S0PC10	S0PC9	S0PC8	R/W	STK0H

✷ 注：
1、为了避免系统错误，在初始化时，请将上表所有寄存器的位都按照设计要求设置为确定的"1"或者"0"；
2、所有寄存器的名称在 SN8ASM 编译器中做了宣告；
3、寄存器中各位的名称已在 SN8ASM 编译器中以“F”为前缀定义过；
4、指令“B0BSET”、“B0BCLR”、“BSET”、“BCLR”只能用于“R/W”寄存器。

2.2.2 累加器ACC

8 位数据寄存器 ACC 用来执行 ALU 与数据存储器之间数据的传送操作。如果操作结果为零（Z）或有进位产生（C 或 DC），程序状态寄存器 PFLAG 中相应位会发生变化。

ACC 并不在 RAM 中，因此在立即寻址模式中不能用“B0MOV”指令对其进行读写。

➢ **例：读/写 ACC。**

```
; 数据写入 ACC。
                MOV         A, #0FH

; 读取 ACC 中的数据并存入 BUF。
                MOV         BUF, A
                B0MOV       BUF, A

; BUF 中的数据写入 ACC。
                MOV         A, BUF
                B0MOV       A, BUF
```

系统执行中断操作时，ACC 和 PFLAG 中的数据不会自动存储，用户需通过程序将中断入口处的 ACC 和 PFLAG 中的数据送入存储器进行保存。可通过“PUSH”和“POP”指令对 ACC 和 PFLAG 等系统寄存器进行存储及恢复。

➢ **例：ACC 和工作寄存器中断保护操作。**

```
INT_SERVICE:
                PUSH                    ; 保存 PFLAG 和 ACC。
                …               .
                …
                POP                     ; 恢复 ACC 和 PFLAG。

                RETI                    ; 退出中断。
```

2.2.3 程序状态寄存器PFLAG

寄存器 PFLAG 中包含 ALU 运算状态信息、系统复位状态信息和 LVD 低电压检测状态信息。其中，位 NT0 和 NPD 显示系统复位状态信息，包括上电复位、LVD 复位、外部复位和看门狗复位；位 C、DC 和 Z 显示 ALU 的运算信息。

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	NT0	NPD	-	-	-	C	DC	Z
读/写	R/W	R/W	-	-	-	R/W	R/W	R/W
复位后	X	X	-	-	-	0	0	0

Bit [7:6] **NT0, NPD**：复位状态标志。

NT0	NPD	复位状态
0	0	看门狗复位
0	1	保留
1	0	LVD 复位
1	1	外部复位

Bit 2 **C**：进位标志。

1 =加法运算后有进位、减法运算没有借位发生或移位后移出逻辑“1”或比较运算的结果≥0；

0 =加法运算后没有进位、减法运算有借位发生或移位后移出逻辑“0”或比较运算的结果<0。

Bit 1 **DC**：辅助进位标志。

1 =加法运算时低四位有进位，或减法运算后没有向高四位借位；

0 =加法运算时低四位没有进位，或减法运算后有向高四位借位。

Bit 0 **Z**：零标志。

1 =算术/逻辑/分支转移运算的结果为零；

0 =算术/逻辑/分支转移运算的结果非零。

✷ **注：关于标志位 C、DC 和 Z 的更多信息请参阅指令集相关内容。**

2.2.4 程序计数器PC

程序计数器 PC 是一个 11 位二进制程序地址寄存器，分高 3 位和低 8 位。专门用来存放下一条需要执行指令的内存地址。通常，程序计数器会随程序中指令的执行自动增加。

若程序执行 CALL 和 JMP 指令时，PC 指向特定的地址。

	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PC	-	-	-	-	-	PC10	PC9	PC8	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
复位后	-	-	-	-	-	0	0	0	0	0	0	0	0	0	0	0
	PCH								PCL							

☞ **单地址跳转**

在 SONiX 单片机里面，有 9 条指令（CMPRS、INCS、INCMS、DECS、DECMS、BTS0、BTS1、B0BTS0 和 B0BTS1）可完成单地址跳转功能。如果这些指令执行结果为真，那么 PC 值加 2 以跳过下一条指令。

如果位检测为真，则跳过下一条指令。

```
                B0BTS1      FC          ; 若 Carry_flag = 1，跳过下一条指令。
                JMP         C0STEP      ; 否则跳到 C0STEP。
                …
C0STEP:         NOP

                B0MOV       A, BUF0     ;
                B0BTS0      FZ          ; 若 Zero flag = 0，跳过下一条指令。
                JMP         C1STEP      ; 否则跳到 C1STEP。
                …
C1STEP:         NOP
```

如果 ACC 等于指定的立即数则 PC 值加 2，跳过下一条指令。

```
                CMPRS       A, #12H     ; 若 ACC = 12H，跳过下一条指令。
                JMP         C0STEP      ; 否则跳到 C0STEP。
                …
C0STEP:         NOP
```

执行加 1 指令后，结果为零时，PC 的值加 2，跳过下一条指令。

INCS：

```
                INCS        BUF0
                JMP         C0STEP      ;
                …
C0STEP:         NOP
```

INCMS：

```
                INCMS       BUF0
                JMP         C0STEP      ;
                …
C0STEP:         NOP
```

执行减 1 指令后，结果为零时，PC 的值加 2，跳过下一条指令。

DECS：

```
                DECS        BUF0
                JMP         C0STEP      ;
                …
C0STEP:         NOP
```

DECMS：

```
                DECMS       BUF0
                JMP         C0STEP      ;
                …
C0STEP:         NOP
```

☞ **多地址跳转**

执行 JMP 或 ADD M,A（M=PCL）指令可实现多地址跳转。执行 ADD M, A、ADC M, A 或 B0ADD M, A 后，若 PCL 溢出，PCH 会自动进位。对于跳转表及其它应用，用户可以通过上述 3 条指令计算 PC 的值而不需要担心 PCL 溢出的问题。

✷ **注：PCH 仅支持 PC 的递增运算而不支持递减运算。当 PCL+ACC 执行完 PCL 有进位时，PCH 会自动加 1；但执行 PCL-ACC 有借位发生，PCH 的值会保持不变。**

➢ **例：PC = 0323H（PCH = 03H，PCL = 23H）。**

```
; PC = 0323H
                MOV         A, #28H
                B0MOV       PCL, A          ; 跳到地址 0328H。
                …

; PC = 0328H
                MOV         A, #00H
                B0MOV       PCL, A          ; 跳到地址 0300H。
                …
```

➢ **例：PC = 0323H（PCH = 03H，PCL = 23H）。**

```
; PC = 0323H
                B0ADD       PCL, A          ; PCL = PCL + ACC，PCH 的值不变。
                JMP         A0POINT         ; ACC = 0，跳到 A0POINT。
                JMP         A1POINT         ; ACC = 1，跳到 A1POINT。
                JMP         A2POINT         ; ACC = 2，跳到 A2POINT。
                JMP         A3POINT         ; ACC = 3，跳到 A3POINT。
                …
                …
```

2.2.5 H, L寄存器

寄存器 H 和 L 都是 8 位缓存器，主要有以下两个功能：

- **通用工作寄存器；**
- **RAM 数据寻址指针@HL。**

081H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
H	HBIT7	HBIT6	HBIT5	HBIT4	HBIT3	HBIT2	HBIT1	HBIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	X	X	X	X	X	X	X	X

080H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
L	LBIT7	LBIT6	LBIT5	LBIT4	LBIT3	LBIT2	LBIT1	LBIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	X	X	X	X	X	X	X	X

➢ **例：用 H、L 作为数据指针，访问 bank0 中 020H 处的内容。**

```
B0MOV   H, #00H
B0MOV   L, #20H
B0MOV   A, @HL
```

➢ **例：对 bank 0 中的数据进行清零处理。**

```
        CLR     H           ; H = 0，指向 bank 0。
        B0MOV   L, #7FH     ; L = 7FH。
CLR_HL_BUF:
        CLR     @HL         ; @HL 清零。
        DECMS   L           ; L – 1，如果 L = 0，程序结束。
        JMP     CLR_HL_BUF

        CLR     @HL
END_CLR:
        …
        …
```

2.2.6 Y, Z寄存器

寄存器 Y 和 Z 都是 8 位缓存器，主要用途如下：

- **普通工作寄存器；**
- **RAM 数据寻址指针@YZ；**
- **配合指令 MOVC 对 ROM 数据进行查表。**

084H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Y	YBIT7	YBIT6	YBIT5	YBIT4	YBIT3	YBIT2	YBIT1	YBIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	X	X	X	X	X	X	X	X

083H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
Z	ZBIT7	ZBIT6	ZBIT5	ZBIT4	ZBIT3	ZBIT2	ZBIT1	ZBIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	X	X	X	X	X	X	X	X

➢ **例：用 Y、Z 作为数据指针，访问 bank0 中 025H 处的内容。**

```
        B0MOV   Y, #00H     ; Y 指向 RAM bank 0。
        B0MOV   Z, #25H     ; Z 指向 25H。
        B0MOV   A, @YZ      ; 数据送入 ACC。
```

➢ **例：利用数据指针@YZ 对 RAM 数据清零。**

```
        B0MOV   Y, #0       ; Y = 0，指向 bank 0。
        B0MOV   Z, #7FH     ; Z = 7FH，RAM 区的最后单元。

CLR_YZ_BUF:
        CLR     @YZ         ; @YZ 清零。

        DECMS   Z           ;
        JMP     CLR_YZ_BUF  ; 不为零。

        CLR     @YZ
END_CLR:                    ;
        …
```

2.2.7 R寄存器

8 位缓存器 R 主要有以下两个功能：

- **作为工作寄存器使用；**
- **存储执行查表指令后的高字节数据。**（执行 MOVC 指令，指定 ROM 单元的高字节数据会被存入 R 寄存器而低字节数据则存入 ACC。）

082H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
R	RBIT7	RBIT6	RBIT5	RBIT4	RBIT3	RBIT2	RBIT1	RBIT0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	X	X	X	X	X	X	X	X

2.3 寻址模式

2.3.1 立即寻址

将立即数直接送入 ACC 或指定的 RAM 单元。

- **例：立即数 12H 送入 ACC。**

```
MOV     A, #12H
```

- **例：立即数 12H 送入寄存器 R。**

```
B0MOV   R, #12H
```

✷ **注：立即寻址模式中，指定的 RAM 单元必须是 80H~87H 的工作寄存器。**

2.3.2 直接寻址

通过 ACC 对 RAM 单元数据进行操作。

- **例：地址 12H 处的内容送入 ACC。**

```
B0MOV   A, 12H
```

- **例：ACC 中数据写入 RAM 中 12H 单元。**

```
B0MOV   12H, A
```

2.3.3 间接寻址

通过数据指针（H/L、Y/Z）对数据存储单元进行读写。

- **例：通过指针@HL 间接寻址。**

```
B0MOV   H, #0       ; 清“H”以寻址 RAM bank 0。
B0MOV   L, #12H     ; 设定寄存器地址。
B0MOV   A, @HL
```

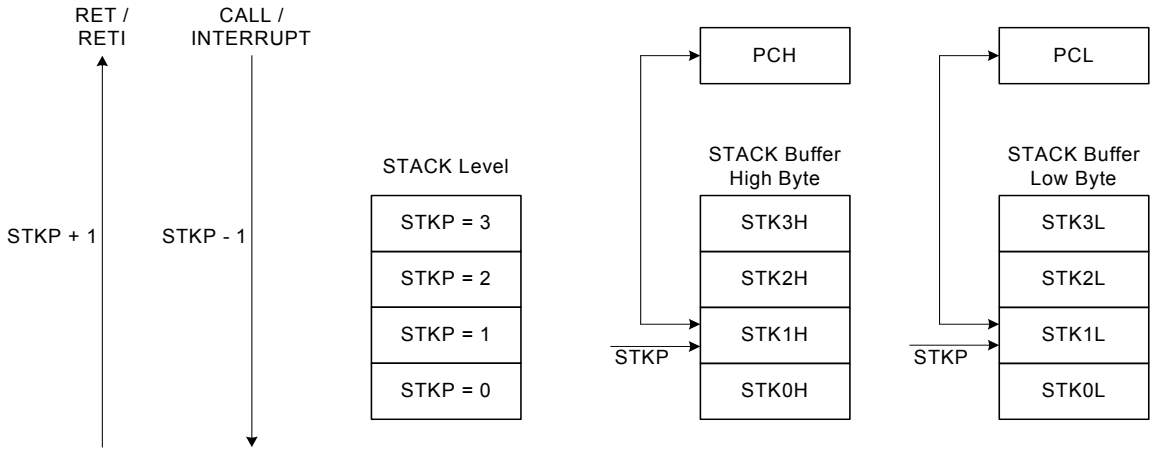
- **例：通过指针@YZ 间接寻址。**

```
B0MOV   Y, #0       ; 清“Y”以寻址 RAM bank 0。
B0MOV   Z, #12H     ; 设定寄存器地址。
B0MOV   A, @YZ
```

2.4 堆栈

2.4.1 概述

SN8PC20 的堆栈缓存器共有 4 层，程序进入中断或执行 CALL 指令时，用来存储程序计数器 PC 的值。寄存器 STKP 为堆栈指针，指向堆栈缓存器顶层，STKnH 和 STKnL 分别是各堆栈缓存器高、低字节。



2.4.2 堆栈寄存器

堆栈指针 STKP 是一个 2 位寄存器，存放被访问的堆栈单元地址，9 位数据存储器 STKnH 和 STKnL 用于暂存堆栈数据。以上寄存器都位于 bank 0。

堆栈操作遵循后进先出（LIFO）的原则，入栈时堆栈指针 STKP 的值减 1，出栈时 STKP 的值加 1，这样，STKP 总是指向堆栈缓存器顶层单元。

系统进入中断或执行 CALL 指令之前，程序计数器 PC 的值被存入堆栈缓存器中进行入栈保护。

0DFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKP	GIE	-	-	-	-	-	STKPB1	STKPB0
读/写	R/W	-	-	-	-	-	R/W	R/W
复位后	0	-	-	-	-	-	1	1

Bit[2:0] **STKPBn：** 堆栈指针（n = 0 ~ 1）。

Bit 7 **GIE:** 全局中断控制位。
0 = 禁止；
1 = 允许。

➢ **例：系统复位时，堆栈指针寄存器内容为默认值，但强烈建议在程序初始部分重新设定，如下面所示：**

```
MOV     A, #00000011B
B0MOV   STKP, A
```

0F0H~0F8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKnH	-	-	-	-	-	SnPC10	SnPC9	SnPC8
读/写	-	-	-	-	-	R/W	R/W	R/W
复位后	-	-	-	-	-	0	0	0

0F0H~0F8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKnL	SnPC7	SnPC6	SnPC5	SnPC4	SnPC3	SnPC2	SnPC1	SnPC0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

STKn = STKnH , STKnL (n = 3 ~ 0)

2.4.3 堆栈操作举例

执行程序调用指令 CALL 和响应中断服务时，堆栈指针 STKP 的值减 1，指针指向下一个堆栈缓存器。同时，对程序计数器 PC 的内容进行入栈保护。入栈操作如下表所示：

堆栈层数	STKP		堆栈缓存器		注释
	STKPB1	STKPB0	高字节	低字节	
0	1	1	Free	Free	-
1	1	0	STK0H	STK0L	-
2	0	1	STK1H	STK1L	-
3	0	0	STK2H	STK2L	-
4	1	1	STK3H	STK3L	-
> 4	1	0	-	-	堆栈溢出

对应每个入栈操作，都有一个出栈操作来恢复程序计数器 PC 的值。RETI 指令用于中断服务程序中，RET 用于子程序调用。出栈时，STKP 加 1 并指向下一个空闲堆栈缓存器。堆栈恢复操作如下表所示：

堆栈层数	STKP		堆栈缓存器		注释
	STKPB1	STKPB0	高字节	低字节	
4	1	1	STK3H	STK3L	-
3	0	0	STK2H	STK2L	-
2	0	1	STK1H	STK1L	-
1	1	0	STK0H	STK0L	-
0	1	1	Free	Free	-

2.5 编译选项列表（CODE OPTION）

编译选项为系统的硬件配置，包括振荡器的类型，看门狗定时器的工作模式，LVD 选项，复位引脚选项和 OTP ROM 的安全控制。其项目如下表所示：

编译选项	配置项目	功能说明
High_Clk	IHRC_8M	高速时钟振荡器采用内部 8MHz RC 振荡器，XIN/XOUT 作为普通的 I/O 引脚。
	RC	外部高速时钟采用廉价的 RC 振荡电路，XOUT 作为普通的 I/O 引脚。
	455K X'tal	外部高速时钟采用低频（32KHz 和 455KHz）、低功耗振荡器。 32KHz：**XIN 引脚连接一个 20uF 的电容，XOUT 引脚连接一个 20pF 的电容。** 455KHz：**XIN 引脚连接一个 47uF 的电容，XOUT 引脚连接一个 20pF 的电容。**
	8M X'tal	外部高速振荡器采用高频的晶体振荡器（如 8MHz）。
	4M X'tal	外部高速振荡器采用标准的晶体振荡器（如 4MHz）。
Fcpu	Fhosc/1	指令周期 = 1 个时钟周期。
	Fhosc/2	指令周期 = 2 个时钟周期。
	Fhosc/4	指令周期 = 4 个时钟周期。
	Fhosc/8	指令周期 = 8 个时钟周期。
Watch_Dog	Always_On	始终开启看门狗定时器，即使在睡眠模式和绿色模式下也处于开启状态。
	Enable	开启看门狗定时器，但在睡眠模式和绿色模式下关闭。
	Disable	关闭看门狗定时器。
Reset_Pin	Reset	使能外部复位引脚。
	P02	P0.2 作为普通的输入引脚，但无上拉电阻。
Security	Enable	ROM 代码加密。
	Disable	ROM 代码不加密。

2.5.1 Reset_Pin编译选项

复位引脚和普通的输入引脚共用，通过编译选项控制工作模式。

- **Reset**：外部复位功能。下降沿触发，系统复位。
- **P02**：单向输入引脚。此时禁止外部复位功能，P0.2 作为普通的输入引脚。

2.5.2 Security编译选项

Security 编译选项保护 OTP ROM。当使能 Security 编译选项时，ROM 的代码处于安全状态，不会被读出。

3 复位

3.1 概述

SN8PC20 有以下几种复位方式：

- **上电复位；**
- **看门狗复位；**
- **LVD 检测复位；**
- **外部复位（仅在外部复位引脚处于使能状态）。**

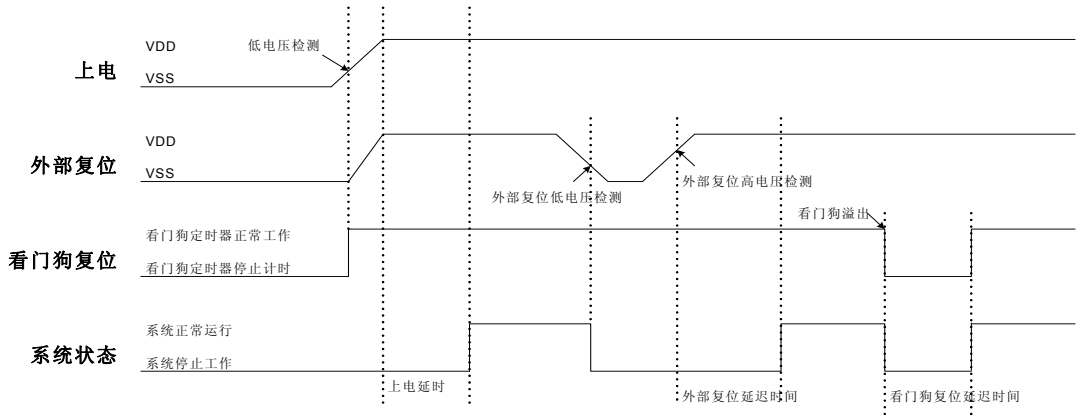
上述任一种复位发生时，所有的系统寄存器恢复默认状态，程序停止运行，同时程序计数器 PC 清零。复位结束后，系统从向量 0000H 处重新开始运行。PFLAG 寄存器的 NT0 和 NPD 两个标志位能够给出系统复位状态的信息。用户可以编程控制 NT0 和 NPD，从而控制系统的运行路径。

086H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PFLAG	NT0	NPD	-	-	-	C	DC	Z
读/写	R/W	R/W	-	-	-	R/W	R/W	R/W
复位后	X	X	-	-	-	0	0	0

Bit [7:6] **NT0, NPD：**复位状态标志位

NT0	NPD	复位情况	说明
0	0	看门狗复位	看门狗溢出
0	1	保留	-
1	0	上电及 LVD 复位	电源电压低于 LVD 检测值
1	1	外部复位	外部复位引脚检测到低电平

任何一种复位情况都需要一定的响应时间，系统提供完善的复位流程以保证复位动作的顺利进行。对于不同类型的振荡器，复位完成所需要的时间也不同。因此，VDD 的上升速度和不同晶振的起振时间都不固定。RC 振荡器的起振时间最短，晶体振荡器的起振时间则较长。在用户使用的过程中，应注意考虑主机对上电复位时间的要求。



3.2 上电复位

上电复位与 LVD 操作密切相关。系统上电过程呈逐渐上升的曲线形式，需要一定时间才能达到正常电平值。下面给出上电复位的正常时序：

- **上电**：系统检测到电源电压上升并等待其稳定；
- **外部复位 (仅限于外部复位引脚使能状态)**：系统检测外部复位引脚状态。如果不为高电平，系统保持复位状态直到
- 外部复位引脚释放。
- **系统初始化**：所有的系统寄存器被置为默认状态；
- **振荡器开始工作**：振荡器开始提供系统时钟；
- **执行程序**：上电结束，程序开始运行。

3.3 看门狗复位

看门狗复位是系统的一种保护设置。在正常状态下，由程序将看门狗定时器清零。若出错，系统处于未知状态，看门狗定时器溢出，此时系统复位。看门狗复位后，系统重启进入正常状态。看门狗复位的时序如下：

- **看门狗定时器状态**：系统检测看门狗定时器是否溢出，若溢出，则系统复位；
- **系统初始化**：初始化所有的系统寄存器；
- **振荡器开始工作**：振荡器开始提供系统时钟；
- **执行程序**：上电结束，程序开始运行。

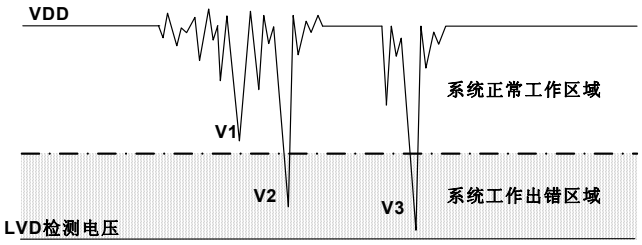
看门狗定时器运用注意事项：

- 看门狗定时器清零之前，请检查 I/O 端口状态及 RAM 数据；
- 不能在中断向量和中断程序中将看门狗定时器清零，否则无法起到侦测程序跑飞的目的；
- 程序中应该只在主程序中有一个清看门狗的动作，这种架构能够最大限度的发挥看门狗的保护功能。

✷ **注：关于看门狗定时器的详细内容，请参阅“看门狗定时器有关章节”。**

3.4 掉电复位

掉电复位针对外部因素引起的系统电压跌落情形（例如，干扰或外部负载的变化），掉电复位可能会引起系统工作状态不正常或程序执行错误。



掉电复位示意图

电压跌落可能会进入系统死区。系统死区意味着电源不能满足系统的最小工作电压要求。上图是一个典型的掉电复位示意图。图中，VDD 受到严重的干扰，电压值降的非常低。虚线以上区域系统正常工作，在虚线以下的区域内，系统进入未知的工作状态，这个区域称作死区。当 VDD 跌至 V1 时，系统仍处于正常状态；当 VDD 跌至 V2 和 V3 时，系统进入死区，则容易导致出错。以下情况系统可能进入死区：

DC 运用中：

DC 运用中一般都采用电池供电，当电池电压过低或单片机驱动负载时，系统电压可能跌落并进入死区。这时，电源不会进一步下降到 LVD 检测电压，因此系统维持在死区。

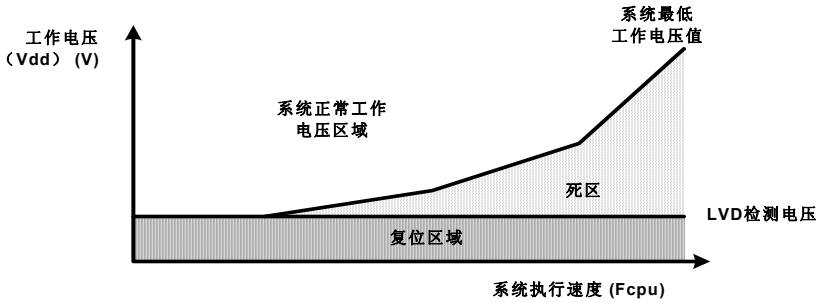
AC 运用中：

系统采用 AC 供电时，DC 电压值受 AC 电源中的噪声影响。当外部负载过高，如驱动马达时，负载动作产生的干扰也影响到 DC 电源。VDD 若由于受到干扰而跌落至最低工作电压以下时，则系统将有可能进入不稳定工作状态。

在 AC 运用中，系统上、下电时间都较长。其中，上电时序保护使得系统正常上电，但下电过程却和 DC 运用中情形类似，AC 电源关断后，VDD 电压在缓慢下降的过程中易进入死区。

3.4.1 系统工作电压

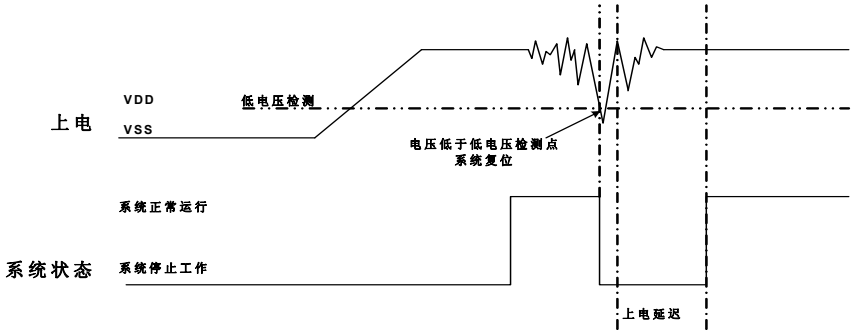
为了改善系统掉电复位的性能，首先必须明确系统具有的最低工作电压值。系统最低工作电压与系统执行速度有关，不同的执行速度下最低工作电压值也不同。



系统工作电压与执行速度关系图

如上图所示，系统正常工作电压区域一般高于系统复位电压，同时复位电压由低电压检测（LVD）电平决定。当系统执行速度提高时，系统最低工作电压也相应提高，但由于系统复位电压是固定的，因此在系统最低工作电压与系统复位电压之间就会出现一个电压区域，系统不能正常工作，也不会复位，这个区域即为死区。

3.4.2 低电压检测（LVD）



低电压检测（LVD）是 SONiX 8 位单片机内置的掉电复位保护装置，当 VDD 跌落并低于 LVD 检测电压值时，LVD 被触发，系统复位。不同的单片机有不同的 LVD 检测电平，LVD 检测电平值仅为一个电压点，并不能覆盖所有死区范围。因此采用 LVD 依赖于系统要求和环境状况。电源变化较大时，LVD 能够起到保护作用，如果电源变化触发 LVD，系统工作仍出错，那么 LVD 就不能起到保护作用，就需要采用其它复位方法。

3.4.3 掉电复位性能改进

如何改善系统掉电复位性能，有以下几点建议：

- **LVD 复位；**
- **看门狗复位；**
- **降低系统工作速度；**
- **采用外部复位电路 (稳压二极管复位电路，电压偏移复位电路，外部 IC 复位)。**

✷ **注：“稳压二极管复位电路”、“电压偏移复位电路” 和 “外部 IC 复位”能够完全避免掉电复位出错。**

看门狗复位：

看门狗定时器用于保证系统正常工作。通常，会在主程序中将看门狗定时器清零，但不要在多个分支程序中清看门狗。若程序正常运行，看门狗不会复位。当系统进入死区或程序运行出错的时候，看门狗定时器继续计数直至溢出，系统复位。

如果看门狗复位后电源仍处于死区，则系统复位失败，保持复位状态，直到系统工作状态恢复到正常值。

降低系统工作速度：

系统工作速度越快最低工作电压值越高，从而加大工作死区的范围，因此降低系统工作速度不失为降低系统进入死区几率的有效措施。所以，可选择合适的工作速度以避免系统进入死区，这个方法需要调整整个程序使其满足系统要求。

附加外部复位电路：

外部复位也能够完全改善掉电复位性能。有三种外部复位方式可改善掉电复位性能：稳压二极管复位电路，电压偏移复位电路和外部 IC 复位。它们都采用外部复位信号控制单片机可靠复位。

3.5 外部复位

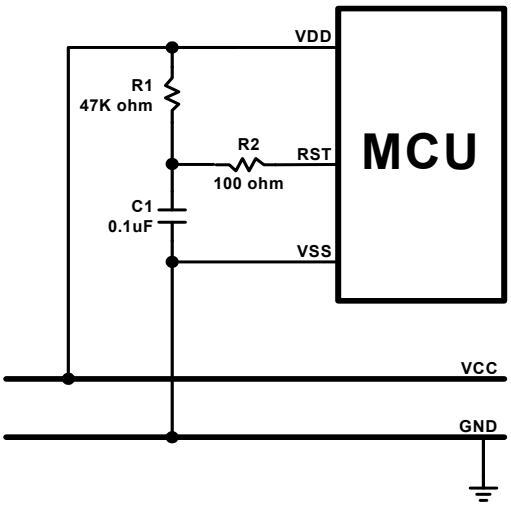
外部复位功能由编译选项“Reset_Pin”控制。将该编译选项置为“Reset”，可使能外部复位功能。外部复位引脚为施密特触发结构，低电平有效。复位引脚处于高电平时，系统正常运行。当复位引脚输入低电平信号时，系统复位。外部复位操作在上电和正常工作模式时有效。需要注意的是，在系统上电完成后，外部复位引脚必须输入高电平，否则系统将一直保持在复位状态。外部复位的时序如下：

- **外部复位（当且仅当外部复位引脚为使能状态）：**系统检测复位引脚的状态，如果复位引脚不为高电平，则系统会一直保持在复位状态，直到外部复位结束；
- **系统初始化：**初始化所有的系统寄存器；
- **振荡器开始工作：**振荡器开始提供系统时钟；
- **执行程序：**上电结束，程序开始运行。

外部复位可以在上电过程中使系统复位。良好的外部复位电路可以保护系统以免进入未知的工作状态，如 AC 应用中的掉电复位等。

3.6 外部复位电路

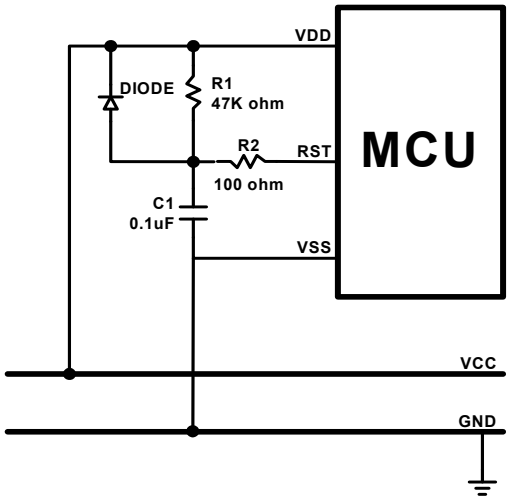
3.6.1 RC复位电路



上图为一个由电阻 R1 和电容 C1 组成的基本 RC 复位电路，它在系统上电的过程中能够为复位引脚提供一个缓慢上升的复位信号。这个复位信号的上升速度低于 VDD 的上电速度，为系统提供合理的复位时序，当复位引脚检测到高电平时，系统复位结束，进入正常工作状态。

✷ **注：此 RC 复位电路不能解决非正常上电和掉电复位问题。**

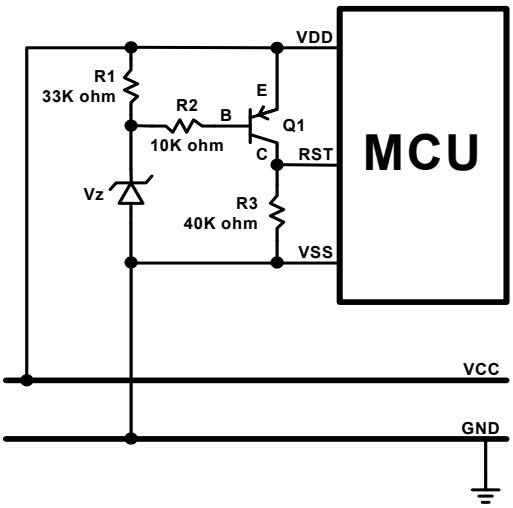
3.6.2 二极管及RC复位电路



上图中，R1 和 C1 同样是为复位引脚提供输入信号。对于电源异常情况，二极管正向导通使 C1 快速放电并与 VDD 保持一致，避免复位引脚持续高电平、系统无法正常复位。

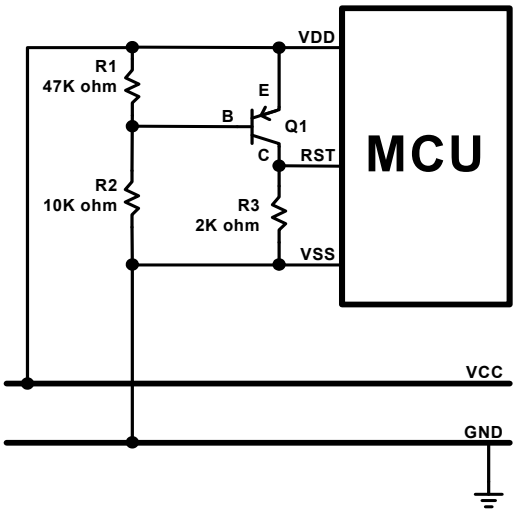
✷ **注：“基本 RC 复位电路”和“二极管及 RC 复位电路”中的电阻 R2 都是必不可少的限流电阻，以避免复位引脚 ESD（Electrostatic Discharge）或 EOS（Electrical Over-stress）)击穿。**

3.6.3 稳压二极管复位电路



稳压二极管复位电路是一种简单的 LVD 电路，基本上可以完全解决掉电复位问题。如上图电路中，利用稳压管的击穿电压作为电路复位检测值，当 VDD 高于“Vz + 0.7V”时，三极管集电极输出高电平，单片机正常工作；当 VDD 低于“Vz + 0.7V”时，三极管集电极输出低电平，单片机复位。稳压管规格不同则电路复位检测值不同，根据电路的要求选择合适的二极管。

3.6.4 电压偏置电路

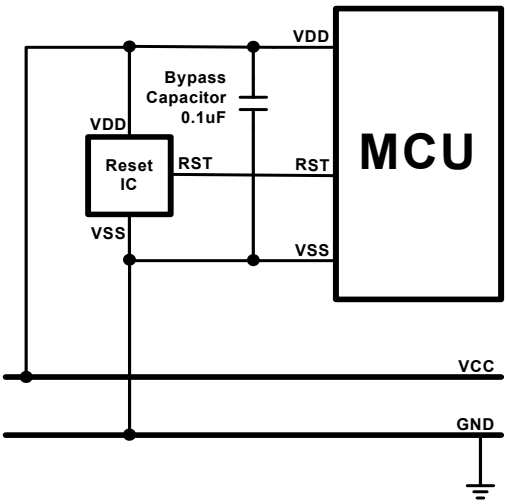


电压偏置复位电路是一种简单的 LVD 电路，基本上可以完全解决掉电复位问题。与稳压二极管复位电路相比，这种复位电路的检测电压值的精确度有所降低。电路中，R1 和 R2 构成分压电路，当 VDD 高于和等于分压值“0.7V x（R1 + R2）/ R1”时，三极管集电极 C 输出高电平，单片机正常工作；VDD 低于“0.7V x（R1 + R2）/ R1”时，集电极 C 输出低电平，单片机复位。

对于不同应用需求，选择适当的分压电阻。单片机复位引脚上电压的变化与 VDD 电压变化之间的差值为 0.7V。如果 VDD 跌落并低于复位引脚复位检测值，那么系统将被复位。如果希望提升电路复位电平，可将分压电阻设置为 R2>R1，并选择 VDD 与集电极之间的结电压高于 0.7V。分压电阻 R1 和 R2 在电路中要耗电，此处的功耗必须计入整个系统的功耗中。

✷ **注：在电源不稳定或掉电复位的情况下，“稳压二极管复位电路”和“偏压复位电路”能够保护电路在电压跌落时避免系统出错。当电压跌落至低于复位检测值时，系统将被复位。从而保证系统正常工作。**

3.6.5 外部IC复位



也可以选用 IC 进行外部复位，但是这样一来系统成本将会增加。针对不同的应用要求选择适当的复位 IC，如上图所示外部 IC 复位电路，能够有效的降低电源变化对系统的影响。

4 系统时钟

4.1 概述

SN8PC20 内置双时钟系统：高速时钟和低速时钟。高速时钟由外部晶振或内部高速振荡电路提供，由编译选项“High_CLK”控制。低速时钟由内置的低速 RC 振荡电路提供，由 OSCM 寄存器的 CLKMD 控制。两种时钟都可作为系统时钟源 Fosc，系统工作在低速模式时，Fosc 4 分频后作为一个指令周期。

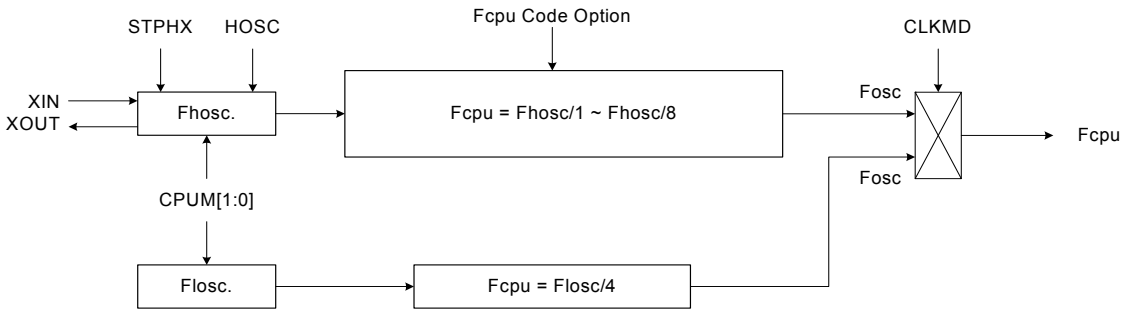
- **高速振荡器**

内部高速振荡器：8MHz RC，称为 IHRC。
外部高速振荡器：包括晶体/陶瓷（4MHz、8MHz 和 455Khz）以及 RC。

- **低速振荡器**

内部低速振荡器：RC 10KHz@3V，称为 ILRC。

- **系统时钟框图**



- HOSC：High_Clk 编译选项。
- Fhosc：外部高速时钟/内部高速 RC 时钟频率。
- Flosc：内部低速 RC 时钟频率（10KHz@3V）。
- Fosc：系统时钟频率。
- Fcpu：指令执行频率。

4.2 FCPU（指令周期）

系统时钟速率，即指令周期（Fcpu），从系统时钟源分离出来，决定系统的工作速率。Fcpu 的速率由 Fcpu 编译选项决定，正常模式下，Fcpu=Fhosc/1~Fhosc/8。当系统高速时钟源由外部 4MHz 晶振提供时，Fcpu 编译选项选择 Fhosc/4，则 Fcpu 频率为 4MHz/4=1MHz。低速模式下，Fcpu=Flosc/4，即 10KHz/4=2.5KHz@3V。

在高干扰环境下，强烈建议 Fcpu=Fhosc/4，以减少高频干扰。

4.3 系统高速时钟

系统高速时钟包括外部高速时钟和内部高速时钟。外部高速时钟又包括 4MHz、8MHz、455KHz 晶体/陶瓷和 RC 振荡器，高速时钟振荡器由编译选项 High_CLK 选择。

4.3.1 HIGH_CLK编译选项

对应不同的时钟功能，SONIX 提供多种高速时钟选项，由 High_CLK 选项控制。High_CLK 选项可以选择 IHRC_8M、8RC、455K X'tal、8M X'tal 和 4M X'tal，以支持不同带宽的振荡器。

- **IHRC_8M**：系统高速时钟源来自内部高速 8MHz RC 振荡器，XIN/XOUT 作为普通的 I/O 引脚，不连接任何外部中断设备。
- **RC**：系统高速时钟源来自廉价的 RC 振荡电路，RC 振荡电路只需要和 XIN 引脚连接，XOUT 作为普通的 I/O 引脚。
- **455K X'tal**：系统高速时钟源来自外部低频振荡器。该选项支持 32KHz 和 455KHz 晶体振荡器。
- **8M X'tal**：系统高速时钟源来自外部高频晶体/陶瓷振荡器，其带宽为 4MHz~8MHz。
- **4M X'tal**：系统高速时钟源来自外部高频晶体/陶瓷振荡器，其带宽为 1MHz~4MHz。

4.3.2 内部高速RC振荡器（IHRC）

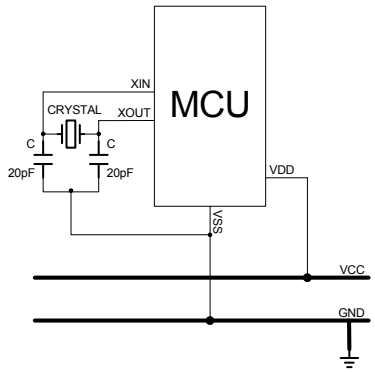
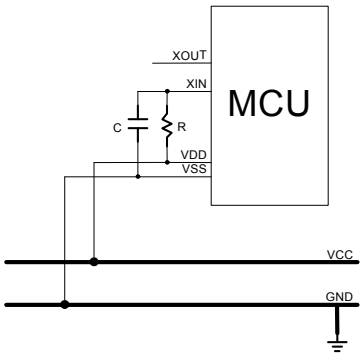
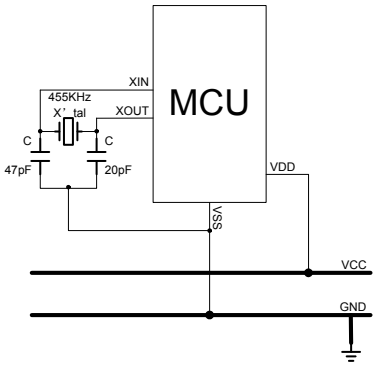
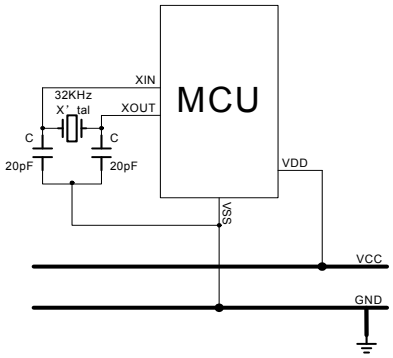
内部高速 8MHz RC 振荡器，普通环境下精确度为±2%，当选择 IHRC_8M 时，使能内部高速振荡器。

- **IHRC_8M**：系统高速时钟为内部 8MHz RC 振荡器，XIN/XOUT 为普通 I/O 引脚。

4.3.3 外部高速振荡器

外部高速振荡器包括 4MHz，8MHz，455KHz 和 RC。4M、8M 和 455K 可以使用晶体和陶瓷振荡器，XIN/XOUT 和 GND 之间需连接一个 20pF 的电容（**455KHz 振荡时，XIN 连接 47pF 的电容，XOUT 连接 20pF 的电容**）。廉价的 RC 振荡电路只需要和 XIN 引脚连接，电容的容值不能低于 100pF，电阻的阻值由频率决定。

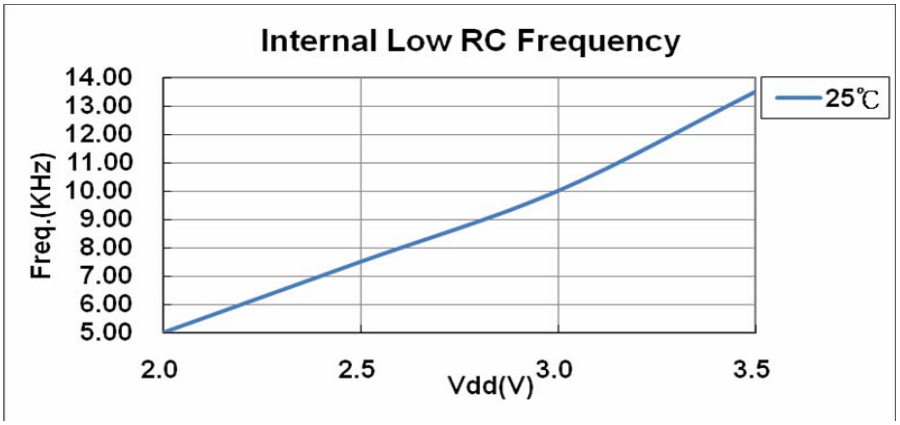
4.3.4 外部振荡应用电路

4M/8M 晶体/陶瓷	RC
	
455KHz 晶体/陶瓷	**32KHz 晶体/陶瓷**
 **455K一和 XIN 引脚连接的电容为 47pF，和 XOUT 引脚连接的电容为 20pF。**	 **455K一和 XIN 引脚连接的电容为 20pF，和 XOUT 引脚连接的电容为 20pF。**

✷ **注：晶体/陶瓷和电容 C 要尽可能的靠近单片机的 XIN/XOUT/VSS；电阻 R 和电容 C 要尽可能的靠近单片机的 VDD。**

4.4 系统低速时钟

系统低速时钟源即内置的低速振荡器，采用 RC 振荡电路。低速时钟的输出频率受系统电压和环境温度的影响，通常为 3V 时输出 10KHZ。输出频率与工作电压之间的关系如下图所示。



低速时钟可作为看门狗定时器的时钟源。由 CLKMD 控制系统低速工作模式。

- **Flosc = 内部低速 RC 振荡器 (10KHz @3V).**
- **低速模式 Fcpu = Flosc / 4**

系统工作在睡眠模式时，可以停止低速 RC 振荡器。

➢ **例：停止内部低速振荡器。**

```
B0BSET      FCPUM0
```

✷ **注：不可以单独停止内部低速时钟；由寄存器 OSCM 的位 CPUM0 和 CPUM1（455K，禁止看门狗）的设置决定内部低速时钟的状态。**

4.5 OSCM寄存器

寄存器 OSCM 控制振荡器的状态和系统模式。

0CAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
OSCM	-	-	-	CPUM1	CPUM0	CLKMD	STPHX	-
读/写	-	-	-	R/W	R/W	R/W	R/W	-
复位后	-	-	-	0	0	0	0	-

Bit 1 **STPHX**：高速振荡器控制位。
0 = 运行；
1 = 停止。内部低速 RC 振荡器仍然运行。

Bit 2 **CLKMD**：高/低速时钟模式控制位。
0 = 普通（双时钟）模式，系统时钟来自高速时钟；
1 = 低速模式，系统时钟来自低速时钟。

Bit[4:3] **CPUM[1:0]**：CPU 工作模式控制位。
00 = 普通模式；
01 = 睡眠模式；
10 = 绿色模式；
11 = 系统保留。

STPHX 位控制内部高速 RC 振荡器和外部振荡器，当 STPHX=0 时，外部振荡器或者内部高速 RC 振荡器有效，STPHX=1 时，外部振荡器或者内部高速 RC 振荡器无效。SPTHX 的功能跟随不同的时钟选项而做不同的控制。

- **IHRC_8M**：STPHX=1，禁止内部高速 RC 振荡器；
- **RC、4M、8M、455K**：STPHX=1，禁止外部振荡器。

4.6 系统时钟测量

在设计过程中，用户可通过软件指令周期 Fcpu 对系统时钟速度进行测试。

➢ **例：外部振荡器的 Fcpu 指令周期测试。**

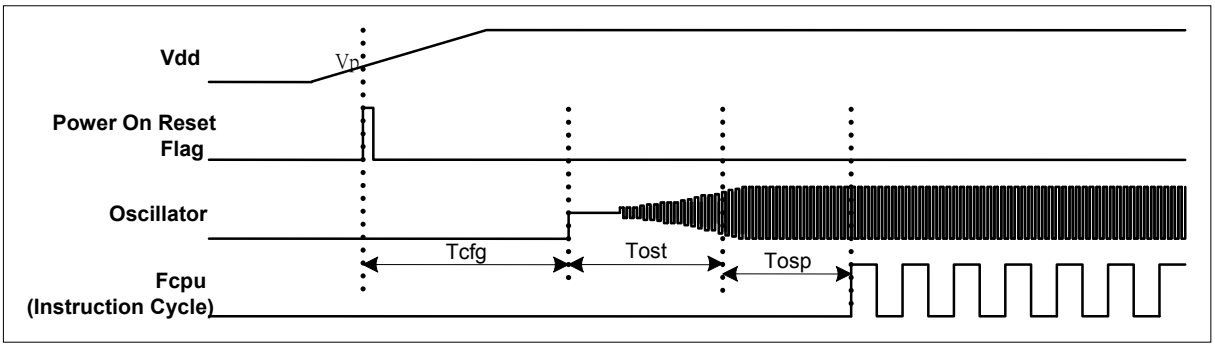
```
        B0BSET      P0M.0           ; P0.0 置为输出模式以输出 Fcpu 的触发信号。
@@:
        B0BSET      P0.0
        B0BCLR      P0.0
        JMP         @B
```

✷ **注：不能直接从 XIN 引脚测试 RC 振荡频率，因为探针的连接会影响测试的准确性。**

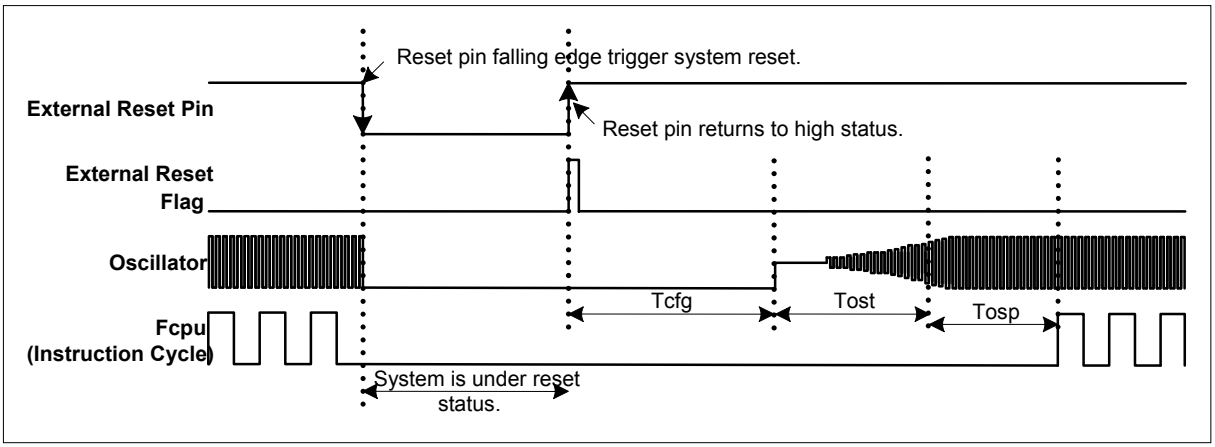
4.7 系统时钟时序

参数	符号	说明	典型值
硬件配置时间	Tcfg	2048*FILRC	64ms @ FILRC = 32KHz 128ms @ FILRC = 16KHz
振荡器启动时间	Tost	启动时间取决于振荡器的材料、工艺等。通常情况下，低速振荡器的启动时间要比高速振荡器的启动时间短，RC 振荡器的启动时间要比晶体/陶瓷振荡器的启动时间短。	-
振荡器起振时间	Tosp	复位情况下的振荡器起振时间为 2048*Fhosc（使能上电复位，LVD 复位，看门狗复位，外部复位引脚）	64ms @ Fhosc = 32KHz 512us @ Fhosc = 4MHz 256us @ Fhosc = 8MHz
		睡眠模式唤醒情况的振荡器起振时间为： 2048*Fhosc ……晶体/陶瓷振荡器，如 32768Hz 晶振，4MHz 晶振，16MHz 晶振等； 32*Fhosc……RC 振荡器，如外部 RC 振荡电路，内部高速 RC 振荡器。	64ms @ Fhosc = 32KHz 512us @ Fhosc = 4MHz 256us @ Fhosc = 8MHz

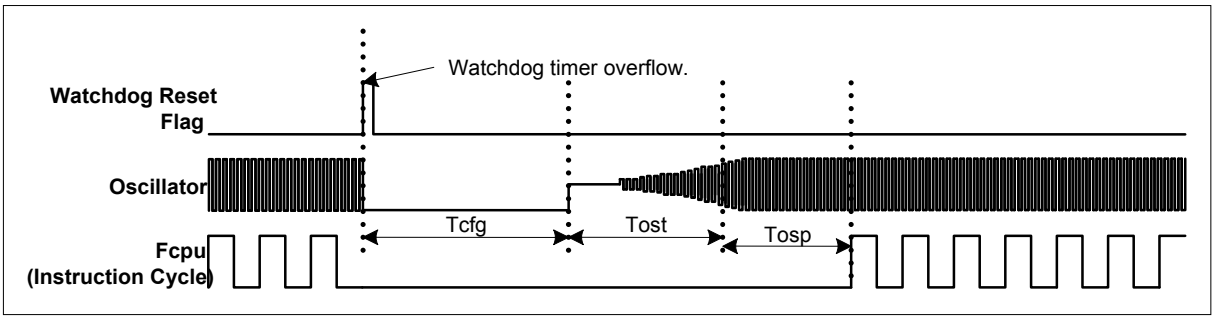
- 上电复位时序



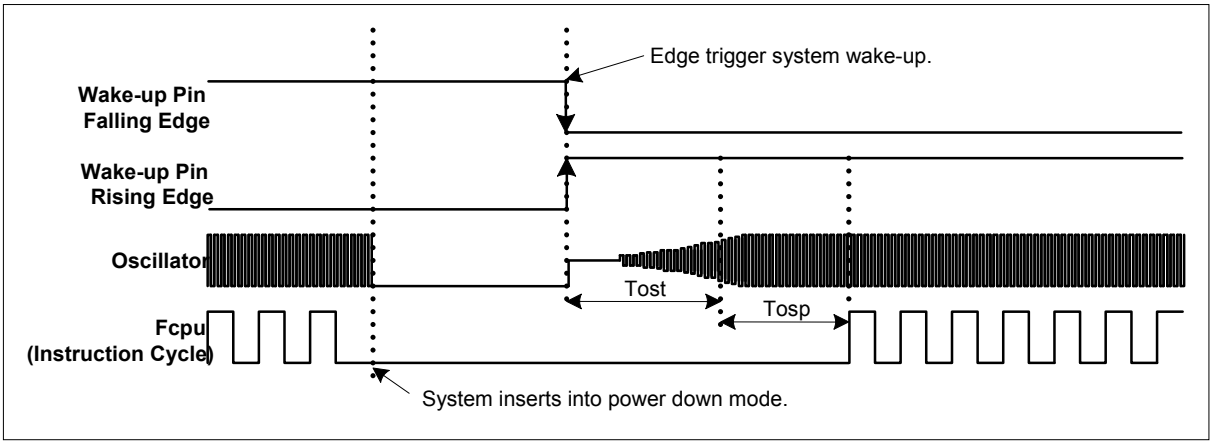
- 外部复位引脚复位时序



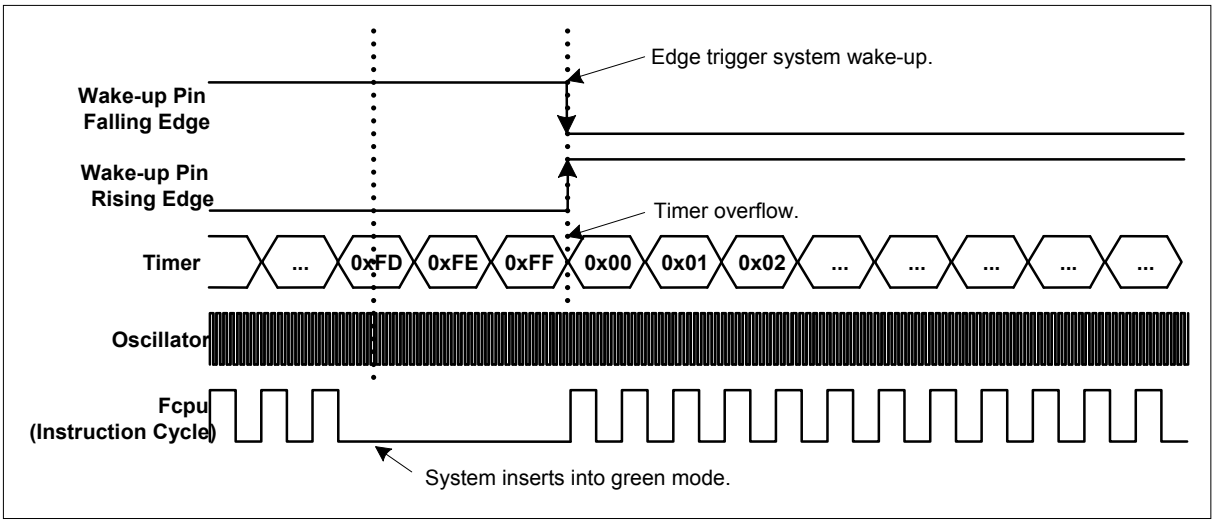
- 看门狗复位时序



- 睡眠模式唤醒时序

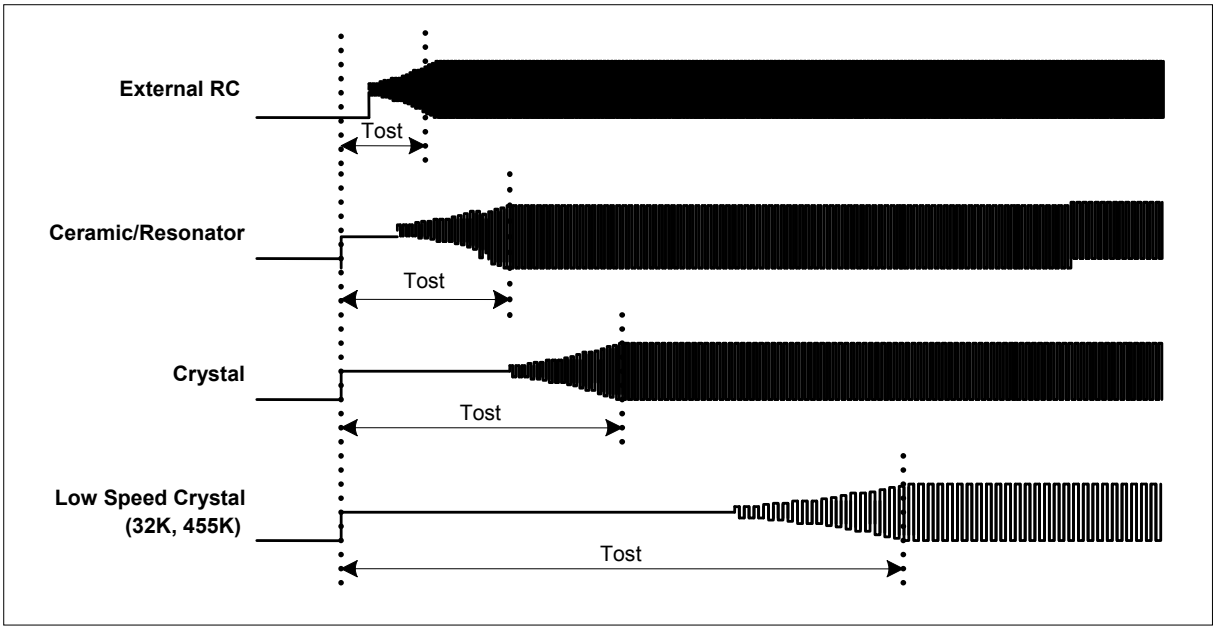


- 绿色模式唤醒时序



- 振荡器启动时间

启动时间取决于振荡器的材料、工艺等。通常情况下，低速振荡器的启动时间要比高速振荡器的启动时间短，RC 振荡器的启动时间要比晶体/陶瓷振荡器的启动时间短。



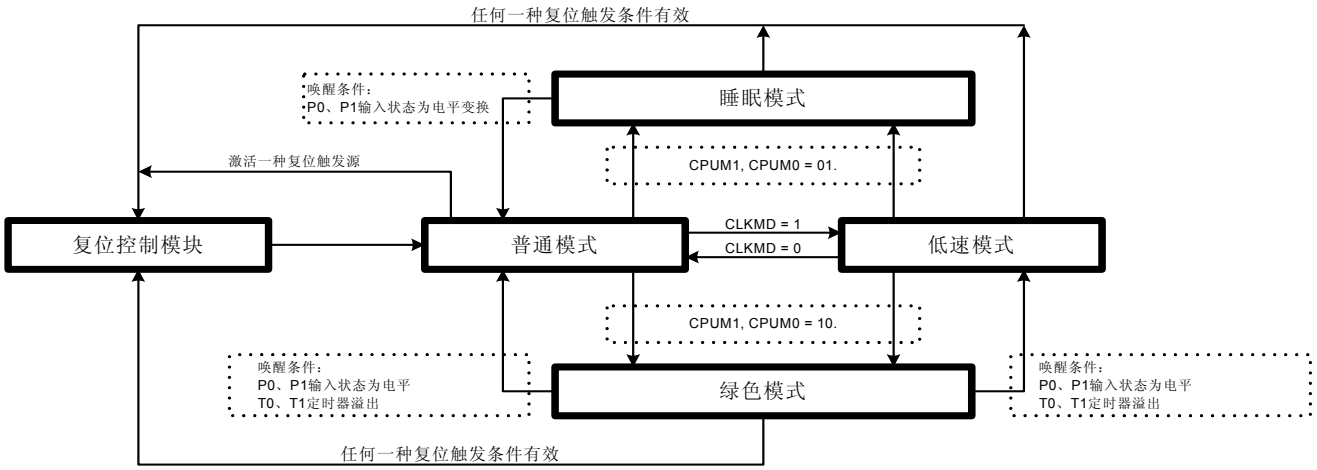
5 系统操作模式

5.1 概述

SN8PC20 可以在 4 种工作模式下以不同的时钟频率工作，这些模式可以控制振荡器，OP 代码操作和模拟设备的操作。

- 普通模式：系统高速工作模式；
- 低速模式：系统低速工作模式；
- 省电模式：系统省电模式（睡眠模式）；
- 绿色模式：系统理想模式。

工作模式控制框图



工作模式时钟控制表

工作模式	普通模式	低速模式	绿色模式	睡眠模式
EHOSC	运行	STPHX	STPHX	停止
IHRC	运行	STPHX	STPHX	停止
ILRC	运行	运行	运行	停止
CPU 指令	执行	执行	停止	停止
T0 定时器	T0ENB	T0ENB	T0ENB	无效
看门狗定时器	Watch_Dog 编译选项	Watch_Dog 编译选项	Watch_Dog 编译选项	Watch_Dog 编译选项
内部中断	全部有效	全部有效	T0	全部无效
外部中断	全部有效	全部有效	全部有效	全部无效
唤醒功能	-	-	P0, P1, T0, 复位	P0, P1, 复位

5.2 普通模式

普通模式即高速模式，系统时钟源由高速振荡器提供，程序正常运行。上电后或者复位后，系统进入普通模式执行程序。当系统从睡眠模式中唤醒后，返回到普通模式。普通模式下，高速振荡器正常工作，因此高速模式也是所有工作模式中功耗最大的一种模式。

- 程序执行时，所有功能可编程控制。
- 系统的执行速率是高速的。
- 系统高速振荡器和内部低速 RC 振荡器都正常运行。
- 通过 OSCM 寄存器可将系统从普通模式切换到其它模式。
- 系统从睡眠模式唤醒后返回到普通模式。
- 低速模式可切换到普通模式。
- 系统从绿色模式唤醒后返回到普通模式。

5.3 低速模式

低速模式下只有低速时钟工作。系统时钟源由内部低速 RC 振荡器（10KHz @3V）提供。低速模式由 OSCM 寄存器的 CLKMD 位控制。当 CLKMD=0 时，系统在普通模式下工作；而当 CLKMD=1 时，系统进入低速模式下工作。此时，高速时钟不会自动被禁止，必须由 SPTHX 位禁止以降低功耗。低速模式下，系统速率被固定在 Flosc/4（Flosc 为内部低速 RC 振荡器频率）。

- 程序执行时，所有功能可编程控制。
- 系统的执行速率是低速的（Flosc/4）。
- 内部低速 RC 振荡器有效，高速振荡器由 STPHX=1 控制。低速模式下，强烈建议停止高速振荡器。
- 通过 OSCM 寄存器可将系统从低速模式切换到其它工作模式。
- 系统从低速模式切换到睡眠模式，被唤醒后进入普通模式。
- 系统可以从普通模式切换进入低速模式。
- 系统从低速模式切换到绿色模式，被唤醒后进入低速模式。

5.4 睡眠模式

睡眠模式即省电模式。睡眠模式下，程序停止执行，高、低速振荡器均停止工作。单片机处于低功耗状态（低于 1uA），睡眠模式由 P0，P1 电平触发唤醒，P1 的唤醒功能由 P1W 寄存器控制。无论系统从何种工作模式进入睡眠模式，被唤醒后都返回到普通模式。用户可以设置 OSCM 寄存器的 CPUM0 位进入睡眠模式。CPUM0=1 时，系统进入睡眠模式。从睡眠模式唤醒后，CPUM0 位（CPUM0=0）自动被禁止。

- 程序停止执行，所有功能被禁止。
- 高、低速振荡器均停止工作。
- 功耗小于 1uA。
- 系统从睡眠模式唤醒后进入普通模式。
- P0 和 P1 电平变化可将系统从睡眠模式中唤醒。

✷ **注：如果系统在普通模式下工作，设置 STPHX=1 可以停止高速时钟振荡器。当所有的振荡器都停止工作时，系统则进入了睡眠模式，可以由 P0，P1 电平变化触发唤醒。**

5.5 绿色模式

绿色模式是不同于睡眠模式的另一种省电模式。睡眠模式下，所有功能被禁止，硬件处于休眠状态。然而在绿色模式下，系统时钟仍处于工作状态，因此绿色模式下功耗会大于睡眠模式下的功耗。绿色模式下，程序停止运行，但需使能具有唤醒功能的定时器，定时器的系统时钟仍然工作。有 2 个唤醒源可以将系统从绿色模式下唤醒：1）P0，P1 的电平变化触发；2）具有唤醒功能的内部定时器溢出触发。也就是说，用户可以给定时器设置一个固定的周期，当定时器溢出时将系统从绿色模式下唤醒。由 OSCM 寄存器的 CPUM1 位决定是否进入绿色模式，当 CPUM1=1 时，系统进入绿色模式。当系统从绿色模式下唤醒后，CPUM1 位（CPUM1=0）自动被禁止。

- 程序停止运行，所有功能被禁止。
- 只有具有唤醒功能的定时器仍然工作。
- 系统时钟振荡器继续运行，另一振荡器的工作状态取决于系统工作模式的设置。
- 系统从普通模式切换进入绿色模式，唤醒后返回到普通模式。
- 系统从低速模式切换进入绿色模式，唤醒后返回到低速模式。
- P0，P1 的电平变化和定时器溢出可将系统从绿色模式中唤醒。

✷ **注：Sonix 提供的宏“GreenMode”可直接进入绿色模式。该宏包括 3 条指令，在使用分支转移指令（bts0, bts1, b0bts0, b0bts1, ins, incms, decs, decms, cmprs, jmp）时，用户需注意对该宏的处理，否则有可能出错。**

5.6 操作模式控制宏

Sonix 提供的工作模式控制宏可以很方便的切换系统工作模式。

宏	长度	描述
SleepMode	1-word	系统进入睡眠模式。
GreenMode	3-word	系统进入绿色模式。
SlowMode	2-word	系统进入低速模式，并停止高速振荡器。
Slow2Normal	5-word	系统从低速模式返回到普通模式。该宏可以控制包括系统模式的切换，使能高速振荡器，高速振荡器起振的延时时间。

➢ **例：系统由普通/低速模式转换到睡眠模式。**

```
SleepMode                    ; 直接宣告“SleepMode”宏。
```

➢ **例：系统由普通模式转换为低速模式。**

```
SlowMode                     ; 直接宣告“SlowMode”宏。
```

➢ **例：系统由低速模式转换到普通模式 (外部高速振荡器停止工作)。**

```
Slow2Normal                  ; 直接宣告“Slow2Normal”宏。
```

➢ **例：系统由普通模式/低速模式进入绿色模式。**

```
GreenMode                    ; 直接宣告“GreenMode”宏。
```

➢ **例：系统由普通/低速模式进入绿色模式，并开启 T0 唤醒功能。**

```
; 设置 T0 定时器的唤醒功能。
        B0BCLR      FT0IEN      ; 禁止 T0 中断。
        B0BCLR      FT0ENB      ; 关闭 T0 定时器。
        MOV         A,#20H      ;
        B0MOV       T0M,A       ; T0 时钟 = Fcpu / 64。
        MOV         A,#64H
        B0MOV       T0C,A       ; 设置 T0C 初始值= 64H（T0 中断间隔 = 10 ms）。
        B0BCLR      FT0IEN      ; 禁止 T0 中断。
        B0BCLR      FT0IRQ      ; T0 中断请求寄存器清零。
        B0BSET      FT0ENB      ; 开启 T0 定时器。
; 进入绿色模式。
        GreenMode               ; 直接宣告“GreenMode” 宏。
```

5.7 唤醒时间

5.7.1 概述

系统在睡眠模式和绿色模式下并不执行程序。唤醒触发信号可以将系统唤醒进入普通模式或低速模式。唤醒触发信号来自外部触发信号（P0 的电平变化）和内部触发信号（T0 定时溢出）。

- 从睡眠模式唤醒后只能进入普通模式，且将其唤醒的触发只能是外部触发信号（P0 电平变化）；
- 由绿色模式唤醒回到系统前一工作模式（普通模式或低速模式）可以用外部触发或者内部触发。

5.7.2 唤醒时间

系统进入睡眠模式后，高速时钟停止运行。把系统从睡眠模式下唤醒时，单片机需要等待 2048 个外部高速振荡器时钟周期以使振荡电路进入稳定工作状态，等待的这一段时间就称为唤醒时间。唤醒时间结束后，系统才进入到普通模式。

✷ **注：将系统从绿色模式中唤醒是不需要唤醒时间的，因为在绿色模式下高速时钟仍然正常工作。**

唤醒时间计算如下：

唤醒时间 = 1/Fosc * 2048（sec）+ 高速时钟启动时间

✷ **注：高速时钟的启动时间与 VDD 和振荡器类型有关。**

➢ **例：将系统从睡眠模式中唤醒，并设置系统进入普通模式。唤醒时间计算如下。**

唤醒时间= 1/Fosc * 2048 = 0.512 ms (Fosc = 4MHz)
总的唤醒时间 = 0.512 ms + 振荡器启动时间

5.7.3 P1W唤醒功能控制寄存器

在绿色模式和睡眠模式下，有唤醒功能的 I/O 口能够将系统唤醒到普通模式。P0 和 P1 都具有唤醒功能，二者区别在于：P0 的唤醒功能始终有效，而 P1 的唤醒功能则由寄存器 P1W 控制。

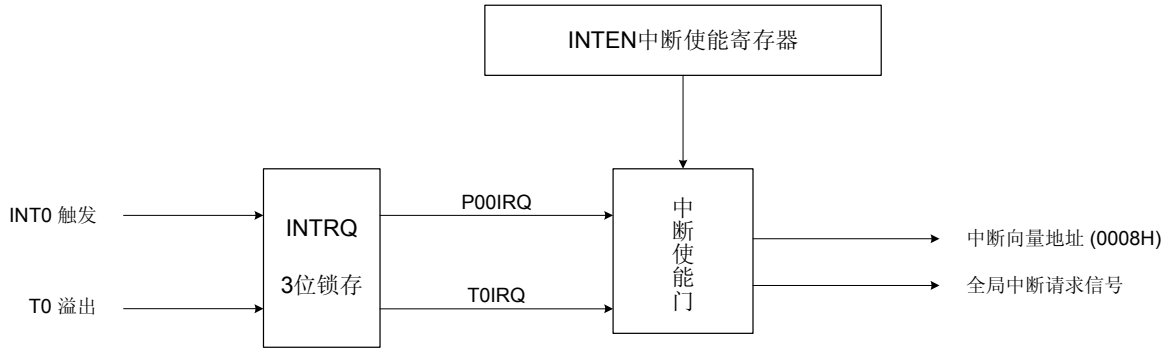
0C0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1W	P17W	P16W	P15W	P14W	P13W	P12W	P11W	P10W
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

Bit[7:0] **P10W~P17W**：P1 唤醒功能控制位。
0 = 禁止；
1 = 使能。

6 中断

6.1 概述

SN8PC20 共有 2 个中断源：1 个内部中断（T0）和 1 个外部中断（INT0）。外部中断可以将系统从睡眠模式唤醒进入高速模式，在返回高速模式前，外部中断请求被锁定。一旦程序进入中断，寄存器 STKP 的位 GIE 将被硬件自动清零以避免再次响应其它中断。系统退出中断，即执行完 RETI 指令后，硬件自动将 GIE 置“1”，以响应下一个中断。中断请求存放在寄存器 INTRQ 中。



✷ **注：程序响应中断时，位 GIE 必须处于有效状态。**

6.2 中断使能寄存器INTEN

中断请求控制寄存器 INTEN 包括所有中断的使能控制位。INTEN 的有效位被置为“1”，则系统进入该中断服务程序，程序计数器入栈，程序转至 0008H 即中断程序。程序运行到指令 RETI 时，中断结束，系统退出中断服务。

0C9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTEN	-	-	-	T0IEN	-	-	-	P00IEN
读/写	-	-	-	R/W	-	-	-	R/W
复位后	-	-	-	0	-	-	-	0

Bit 0 **P00IEN**：P0.0 外部中断（INT0）控制位。
0 = 禁止；
1 = 使能。

Bit 4 **T0IEN**：T0 中断控制位。
0 = 禁止；
1 = 使能。

6.3 中断请求寄存器INTRQ

中断请求寄存器 INTRQ 中存放各中断请求标志。一旦有中断请求发生，INTRQ 中的相应位将被置“1”，该请求被响应后，程序应将该标志位清零。根据 INTRQ 的状态，程序判断是否有中断发生，并执行相应的中断服务。

0C8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
INTRQ	-	-	-	T0IRQ	-	-	-	P00IRQ
读/写	-	-	-	R/W	-	-	-	R/W
复位后	-	-	-	0	-	-	-	0

Bit 0 **P00IRQ**：P0.0 中断（INT0）请求标志位。
0 = INT0 无中断请求；
1 = INT0 有中断请求。

Bit 4 **T0IRQ**：T0 中断请求标志位。
0 = T0 无中断请求；
1 = T0 有中断请求。

6.4 全局中断GIE

只有当全局中断控制位 GIE 置“1”的时候程序才能响应中断请求。 一旦有中断发生，程序计数器（PC）指向中断向量地址（0008H），堆栈层数加 1。

0DFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
STKP	GIE	-	-	-	-	-	STKPB1	STKPB0
读/写	R/W	-	-	-	-	-	R/W	R/W
复位后	0	-	-	-	-	-	1	1

Bit 7 **GIE**：全局中断控制位。
0 = 禁止全局中断；
1 = 使能全局中断。

➢ **例：设置全局中断控制位（GIE）。**

```
B0BSET      FGIE        ; 使能 GIE。
```

✷ **注：在所有中断中，GIE 都必须处于使能状态。**

6.5 PUSH, POP

有中断请求发生并被响应后，程序转至 0008H 执行中断子程序。响应中断之前，必须保存 ACC、PFLAG 的内容。芯片提供 PUSH 和 POP 指令进行入栈保存和出栈恢复，从而避免中断结束后可能的程序运行错误。

✷ **注：“PUSH”、“POP”指令仅对 ACC 和 PFLAG 作中断保护，而不包括 NT0 和 NPD。PUSH/POP 缓存器是唯一的且仅有一层。**

➢ **例：对 ACC 和 PAFLG 进行入栈保护。**

```
                ORG     0
                JMP     START

                ORG     8H
                JMP     INT_SERVICE

                ORG     10H
START:
                …

INT_SERVICE:
                PUSH                    ; 保存 ACC 和 PFLAG。
                …
                …
                POP                     ; 恢复 ACC 和 PFLAG。

                RETI                    ; 退出中断。
                …
                ENDP
```

6.6 外部中断INT0

SN8PC20 提供外部中断 INT0，当外部中断被触发时，若使能外部中断使能控制位，则外部中断请求标志位被置 1。若使能外部中断使能控制位且外部中断请求位置 1 时，程序计数器跳转到中断向量地址 0008H 开始执行中断服务。若禁止外部中断使能控制位，则外部中断请求标志位无效，则不会执行中断服务程序。

外部中断内置唤醒锁存功能，就是指系统从睡眠模式唤醒，唤醒源位中断源（P0.0）。触发沿的方向与中断沿的配置相互配合。当触发沿被锁存后，系统被唤醒后先执行中断服务程序。

0BFH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PEDGE	-	-	-	P00G1	P00G0	-	-	-
读/写	-	-	-	R/W	R/W	-	-	-
复位后	-	-	-	1	0	-	-	-

Bit[4:3] **P00G[1:0]**：P0.0 中断触发控制位。

00 = 保留；
01 = 上升沿触发；
10 = 下降沿触发；
11 = 上升/下降沿触发（电平触发）。

➢ **例：INT0 中断请求设置，电平触发。**

```
        MOV     A, #18H
        B0MOV   PEDGE, A        ; 设置 INT0 为电平触发。

        B0BCLR  FP00IRQ         ; 清 INT0 中断请求标志。
        B0BSET  FP00IEN         ; 使能 INT0 中断。
        B0BSET  FGIE            ; 使能 GIE。
```

➢ **例：INT0 中断。**

```
        ORG     8H              ;
        JMP     INT_SERVICE
INT_SERVICE:

        …                       ; 保存 ACC 和 PFLAG。

        B0BTS1  FP00IRQ         ; 检查是否有 P00 中断请求标志。
        JMP     EXIT_INT        ; P00IRQ = 0，退出中断。

        B0BCLR  FP00IRQ         ; 清 P00IRQ。
        …                       ; INT0 中断服务程序。
        …
EXIT_INT:
        …                       ; 恢复 ACC 和 PFLAG。

        RETI                    ; 退出中断。
```

6.7 T0 中断

T0C 溢出时，无论 T0IEN 处于何种状态，T0IRQ 都会置“1”。若 T0IEN 和 T0IRQ 都置“1”，系统就会响应 T0 的中断；若 T0IEN = 0，则无论 T0IRQ 是否置“1”，系统都不会响应 T0 中断。尤其需要注意多种中断下的情形。

- **例：设置 T0 中断。**

```
        B0BCLR      FT0IEN          ; 禁止 T0 中断。
        B0BCLR      FT0ENB          ; 关闭 T0。
        MOV         A, #20H         ;
        B0MOV       T0M, A          ; 设置 T0 时钟= Fcpu / 64。
        MOV         A, #64H         ; 初始化 T0C = 64H。
        B0MOV       T0C, A          ; 设置 T0 间隔时间= 10 ms。

        B0BCLR      FT0IRQ          ; T0IRQ 清零。
        B0BSET      FT0IEN          ; 使能 T0 中断。
        B0BSET      FT0ENB          ; 开启定时器 T0。

        B0BSET      FGIE            ; 使能 GIE。
```

- **例：T0 中断服务程序。**

```
        ORG         8H
        JMP         INT_SERVICE
INT_SERVICE:

        …                           ; 保存 ACC 和 PFLAG。

        B0BTS1      FT0IRQ          ; 检查是否有 T0 中断请求标志。
        JMP         EXIT_INT        ; T0IRQ = 0，退出中断。

        B0BCLR      FT0IRQ          ; 清 T0IRQ。
        MOV         A, #64H
        B0MOV       T0C, A          ;
        …
        …
EXIT_INT:
        …                           ; 恢复 ACC 和 PFLAG。


        RETI                        ; 退出中断。
```

6.8 多中断操作

在同一时刻，系统中可能出现多个中断请求。此时，用户必须根据系统的要求对各中断进行优先权的设置。中断请求标志 IRQ 由中断事件触发，当 IRQ 处于有效值“1”时，系统并不一定会响应该中断。各中断触发事件如下表所示：

中断	有效触发
P00IRQ	由 PEDGE 控制
T0IRQ	T0C 溢出

多个中断同时发生时，需要注意的是：首先，必须预先设定好各中断的优先权；其次，利用 IEN 和 IRQ 控制系统是否响应该中断。在程序中，必须对中断控制位和中断请求标志进行检测。

➢ **例：多中断条件下检测中断请求。**

```
        ORG     8H
        JMP     INT_SERVICE
INT_SERVICE:
        …                               ; 保存 ACC 和 PFLAG。
INTP00CHK:                              ; 检查是否有 INT0 中断请求。
        B0BTS1  FP00IEN                 ; 检查是否使能 INT0 中断。
        JMP     INTT0CHK                ; 跳到下一个中断。
        B0BTS0  FP00IRQ                 ; 检查是否有 INT0 中断。
        JMP     INTP00                  ; 进入 INT0 中断。
INTT0CHK:                               ; 检查是否有 T0 中断请求。
        B0BTS1  FT0IEN                  ; 检查是否使能 T0 中断。
        JMP     INT_EXIT                ;
        B0BTS0  FT0IRQ                  ; 检查是否有 T0 中断请求。
        JMP     INTT0                   ; 进入 T0 中断。
        …
INT_EXIT:
        …                               ; 恢复 ACC 和 PFLAG。
        RETI                            ; 退出中断。
```

7 I/O口

7.1 概述

SN8PC20 内带 16 个 I/O 引脚。部分 I/O 引脚都和模拟引脚及特殊功能的引脚共用，I/O 引脚的共用情况如下表所示：

I/O 引脚		共用引脚		共用引脚的控制条件
名称	类型	名称	类型	
P0.0	I/O	INT0	DC	P00IEN=1
P0.2	I	RST	DC	Reset_Pin = Reset
		VPP	HV	OTP 烧录
P0.4	I/O	XOUT	AC	High_CLK = 455K，4M，8M
P0.3	I/O	XIN	AC	High_CLK = RC，455K，4M，8M
P5.4	I/O	IROUT	DC	IREN = 1

*** DC：数字特性，AC：模拟特性，HV：高压特性。**

7.2 I/O口模式

寄存器 PnM 控制 I/O 的工作模式。

0B8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0M	P07M	P06M	P05M	P04M	P03M	-	P01M	P00M
读/写	R/W	R/W	R/W	R/W	R/W	-	R/W	R/W
复位后	0	0	0	0	0	-	0	0

0C1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1M	P17M	P16M	P15M	P14M	P13M	P12M	P11M	P10M
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

0C5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5M	-	-	-	P54M	-	-	-	P50M
读/写	-	-	-	R/W	-	-	-	R/W
复位后	-	-	-	0	-	-	-	0

Bit[7:0] **PnM[7:0]**：Pn 模式选择控制位（n = 0~5）。
0 = 输入模式；
1 = 输出模式。

✸ 注：
1、用户可通过位操作指令（B0BSET、B0BCLR）对 I/O 口进行编程控制；
2、P0.2 是单向输入引脚，P0M.2 未定义。

➢ **例：I/O 模式选择。**

```
CLR     P0M         ; 设置为输入模式。
CLR     P1M
CLR     P5M

MOV     A, #0FFH    ; 设置为输出模式。
B0MOV   P0M, A
B0MOV   P1M,A
B0MOV   P5M, A

B0BCLR  P1M.0       ; 设置 P1.0 为输入模式。

B0BSET  P1M.0       ; 设置 P1.0 为输出模式。
```

7.3 I/O口上拉电阻寄存器

I/O 引脚都内置上拉电阻，但只在输入模式时有效。这些上拉电阻由寄存器 PnUR 控制，当 PnUR = 0 时，禁止上拉电阻，而当 PnUR = 1 时，使能上拉电阻。

0E0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0UR	P07R	P06R	P05R	P04R	P03R	-	P01R	P00R
读/写	W	W	W	W	W	-	W	W
复位后	0	0	0	0	0	-	0	0

0E10H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1UR	P17R	P16R	P15R	P14R	P13R	P12R	P11R	P10R
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

0E5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5UR	-	-	-	P54R	-	-	-	P50R
读/写	-	-	-	W	-	-	-	W
复位后	-	-	-	0	-	-	-	0

✷ **注：P0.2 为单向输入引脚，且无上拉电阻，P0UR.2 未定义。**

➢ **例：I/O 口的上拉电阻。**

```
MOV     A, #0FFH        ; 使能 P0、P1 和 P5 的上拉电阻。
B0MOV   P0UR, A         ;
B0MOV   P1UR,A
B0MOV   P5UR, A
```

7.4 I/O口数据寄存器

0D0H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P0	P07	P06	P05	P04	P03	P02	P01	P00
读/写	R/W	R/W	R/W	R/W	R/W	R	R/W	R/W
复位后	0	0	0	0	0	0	0	0

0D1H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P1	P17	P16	P15	P14	P13	P12	P11	P10
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

0D5H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
P5	-	-	-	P54	-	-	-	P50
读/写	-	-	-	R/W	-	-	-	R/W
复位后	-	-	-	0	-	-	-	0

✷ **注：当使能外部复位功能后，P02 保持为“1”。**

➢ **例：读取输入口的数据。**

```
B0MOV   A, P0       ; 读取 P0、P1 和 P5 口的数据。
B0MOV   A, P1
B0MOV   A, P5
```

➢ **例：写入数据到输出端口。**

```
MOV     A, #0FFH    ; 写入数据 FFH 到 P0、P1 和 P5。
B0MOV   P0, A
B0MOV   P1, A
B0MOV   P5, A
```

➢ **例：写入 1 位数据到输出端口。**

```
B0BSET  P1.0        ; P1.0 和 P5.4 置“1”。
B0BSET  P5.4

B0BCLR  P1.0        ; P1.0 和 P5.4 置“0”。
B0BCLR  P5.4
```

8 定时器

8.1 看门狗定时器

看门狗定时器 WDT 是一个 4 位二进制计数器，用于监控程序的正常执行。如果由于干扰，程序进入了未知状态，看门狗定时器溢出，系统复位。看门狗的工作模式由编译选项控制，其时钟源由内部低速 RC 振荡器(10KHz @3V) 提供，它是将内部 RC 振荡器经 512 分频后送往看门狗定时器进行计数。

看门狗溢出时间 = 8192 /内部低速振荡器周期（sec）.

VDD	内部低速 RC 频率	看门狗溢出时间
3V	10KHz	819.2ms

看门狗定时器由编译选项控制，在编译选项（code option）中，用户可以通过 WatchDog 的三个配置选项，选择开启或关闭 WatchDog。

- **Disable**：关闭看门狗定时器。
- **Enable**：开启看门狗定时器。看门狗在普通模式和低速模式时开启，在睡眠模式和绿色模式下关闭。
- **Always_On**：开启看门狗定时器。看门狗始终处于开启状态，睡眠模式和绿色模式下也开启。

在干扰严重的环境下，强烈建议用户选择看门狗定时器“Always_On” 选项，系统出错后会自动复位并重新运行。

看门狗清零的方法是对看门狗计数器清零寄存器 WDTR 写入清零控制字 5AH。

0CCH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
WDTR	WDTR7	WDTR6	WDTR5	WDTR4	WDTR3	WDTR2	WDTR1	WDTR0
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

➢ **例：下面是对看门狗定时器操作的示例程序，在主程序的开始清看门狗定时器。**

```
Main:
        MOV     A, #5AH         ; 清看门狗定时器。
        B0MOV   WDTR, A
        …
        CALL    SUB1
        CALL    SUB2
        …
        JMP     MAIN
```

➢ **例：用宏指令@RST_WDT 清看门狗定时器。**

```
Main:
        @RST_WDT                ; 清看门狗定时器。
        …
        CALL    SUB1
        CALL    SUB2
        …
        JMP     MAIN
```

看门狗定时器应用注意事项如下：

- 对看门狗清零之前，检查 I/O 口的状态和 RAM 的内容可增强程序的可靠性；
- 不能在中断中对看门狗清零，否则无法侦测到主程序跑飞的状况；
- 程序中应该只在主程序中有一次清看门狗的动作，这种架构能够最大限度的发挥看门狗的保护功能。

➢ **例：下面是对看门狗定时器操作的示例程序，在主程序的开始清看门狗定时器。**

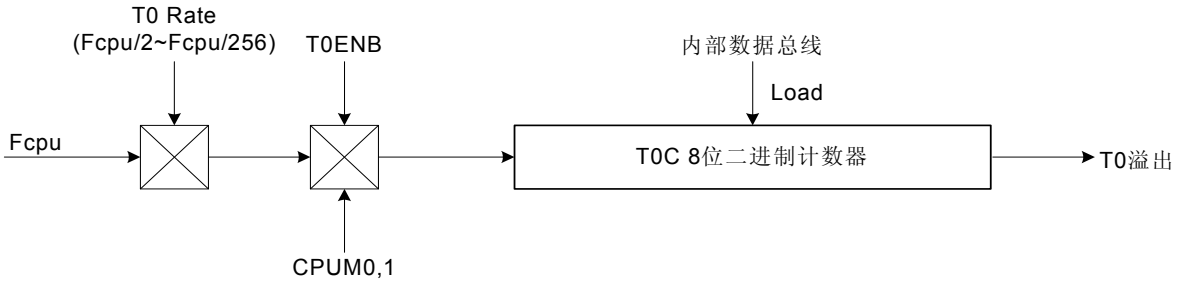
```
Main:
                …                               ; 检查 I/O 口的状态。
                …                               ; 检查 RAM 的内容。
Err:            JMP $                           ; I/O 口或 RAM 出错，不清看门狗等看门狗计时溢出。

Correct:                                        ; I/O 口和 RAM 都正确，清看门狗定时器。
                                                ;
                MOV         A, #5AH             ; 清看门狗定时器。
                B0MOV       WDTR, A
                …
                CALL        SUB1
                CALL        SUB2
                …
                …
                …
                JMP         MAIN
```

8.2 基本定时器T0

8.2.1 概述

8 位二进制基本定时器 T0 具有在绿色模式下唤醒功能。在绿色模式下，T0 溢出，则将系统唤醒返回到上一个工作模式。

- **8 位可编程计数定时器**：根据选择的时钟频率周期性的产生中断请求。
- **中断功能**：T0 定时器支持中断功能，当 T0 溢出，T0IRQ 有效，程序计数器跳到中断向量地址执行中断。
- **绿色模式唤醒功能**：T0 定时器在绿色模式下正常工作，溢出时将系统从绿色模式下唤醒。



8.2.2 T0 操作

T0 定时器的时钟源来自 Fcpu，T0 的单位时间由 T0rate 控制，可以为 Fcpu/2，Fcpu/4，Fcpu/8，Fcpu/16，Fcpu/32，Fcpu/64，Fcpu/128 和 Fcpu/256。当 T0ENB=1 时，T0 开始计数。T0C 溢出（从 0FFH 到 00H）时，T0IRQ 置 1。如果使能 T0 中断（T0IEN=1），T0 溢出后系统执行中断服务程序。T0 可以在绿色模式下将系统唤醒返回到上一个工作模式，T0 没有自动重装功能。可以由程序设置 T0C 来设置 T0 的间隔时间，但必须在 T0 溢出时再次设置，否则 T0 会从 00H 开始计数到 0FFH 共 256 次，而不保持正确的间隔时间。

8.2.3 T0M模式寄存器

0D8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0M	T0ENB	T0rate2	T0rate1	T0rate0	-	-	-	-
读/写	R/W	R/W	R/W	R/W	-	-	-	-
复位后	0	0	0	0	-	-	-	-

Bit [6:4] **T0RATE[2:0]**：T0 分频选择位。

000 = fcpu/256；
001 = fcpu/128；
…；
110 = fcpu/4；
111 = fcpu/2。

Bit 7 **T0ENB**：T0 启动控制位。

0 = 禁止；
1 = 使能。

8.2.4 T0C计数寄存器

8 位计数寄存器 T0C 用于控制 T0 的中断间隔时间。

0D9H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
T0C	T0C7	T0C6	T0C5	T0C4	T0C3	T0C2	T0C1	T0C0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

T0C 初始值的计算公式如下：

T0C 初始值 = 256 -（T0 中断间隔时间 * 输入时钟）

- **例：T0 的中断间隔时间为 10ms，高速时钟为外部 4MHz，Fcpu = Fosc/4，T0RATE = 010（Fcpu/64）。**

  T0C 初始值 = 256 -（T0 中断间隔时间 * 输入时钟）
  = 256 -（10ms * 4MHz / 1 / 64）
  = 256 -（10-2 * 4 * 106 / 1 / 64）
  = 100
  = 64H

T0 的中断间隔时间列表

T0RATE	T0CLOCK	高速模式（Fcpu = 4MHz / 4）		低速模式（Fcpu = 32768Hz / 4）	
		最大间隔溢出时间	单步间隔时间 = max/256	最大间隔溢出时间	单步间隔时间 = max/256
000	Fcpu/256	65.536 ms	256 us	8000 ms	31250 us
001	Fcpu/128	32.768 ms	128 us	4000 ms	15625 us
010	Fcpu/64	16.384 ms	64 us	2000 ms	7812.5 us
011	Fcpu/32	8.192 ms	32 us	1000 ms	3906.25 us
100	Fcpu/16	4.096 ms	16 us	500 ms	1953.125 us
101	Fcpu/8	2.048 ms	8 us	250 ms	976.563 us
110	Fcpu/4	1.024 ms	4 us	125 ms	488.281 us
111	Fcpu/2	0.512 ms	2 us	62.5 ms	244.141 us

8.2.5 T0 定时器操作流程

T0 的操作流程如下：

- **T0 停止计数，禁止 T0 中断功能，并清除 T0 中断请求标志。**

```
B0BCLR    FT0ENB    ;
B0BCLR    FT0IEN    ; 禁止 T0 中断。
B0BCLR    FT0IRQ    ; 清 T0IRQ。
```

- **设置 T0 速率。**

```
MOV       A, #0xxx0000b    ; T0M 的 bit4~bit6 将 T0 的速率控制在 x000xxxxb~x111xxxxb。
B0MOV     T0M,A            ; 关闭 T0 定时器。
```

- **设置 T0 的中断间隔时间。**

```
MOV       A,#7FH
B0MOV     T0C,A            ; 设置 T0C 的值。
```

- **设置 T0 的功能模式。**

```
B0BSET    FT0IEN           ; 开启 T0 的中断功能。
```

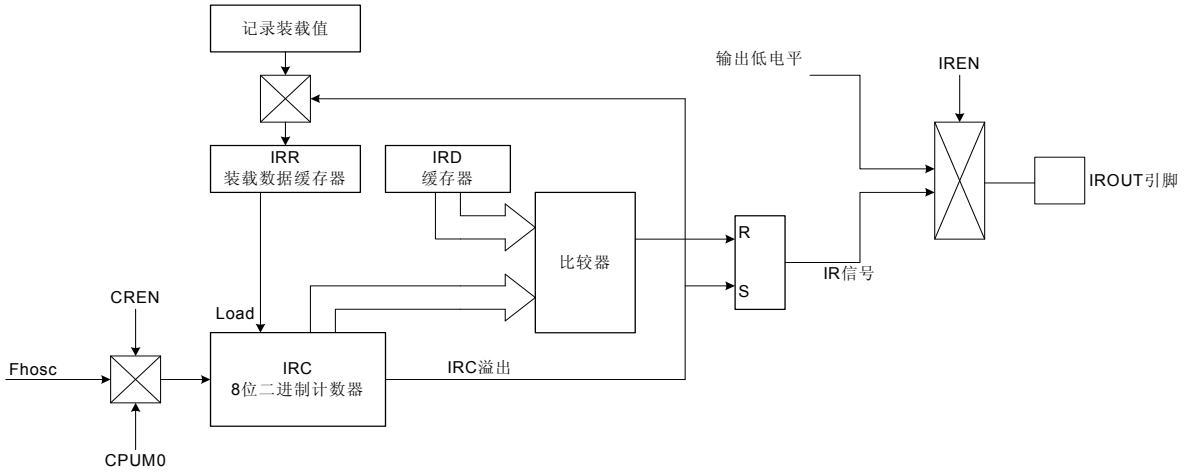
- **开启 T0 定时器。**

```
B0BSET    FT0ENB    ;
```

9 IR输出

9.1 概述

红外信号由 IR 定时器产生，IROUT 引脚最大可以承受 400mA 灌电流。当 IREN 位为“1”时，IROUT 引脚由普通 IO 转变为红外输出模式。当 CREN = 0 或者系统处于睡眠模式时，IROUT 输出高电平。IR 定时器是 8 位二进制累加定时器。由寄存器 IRR 和 IRD 来控制 IR 信号的频率和占空比。IRR 控制红外信号的频率，IRD 控制红外信号的占空比。IR 计数时钟源来自外部高速时钟源（Fhosc），如 IHRC_8M、4MHz 或 455KHz 振荡器。当 Fhosc = 4MHz 时，IR 计数时钟速率就是 4MHz。IR 定时器只能产生红外输出信号，不支持中断功能。当 CREN = 1 时，IR 输出低电平，同时 IRC 开始计数，IRC 的初始值为 IRR。当 IRC = IRD，IR 输出从低电平转变为高电平输出；当 IRC 溢出（0FFH 到 00H）时，IR 输出不再输出高电平。同时系统自动将 IRR 中的值装载到 IRC 寄存器中并进入下一个循环周期。



9.2 IR控制寄存器

9.2.1 IRM模式寄存器

0DAH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IRM	-	-	-	-	-	-	IREN	CREN
读/写	-	-	-	-	-	-	R/W	R/W
复位后	-	-	-	-	-	-	0	0

Bit 1 **IREN**：IROUT 引脚输出控制位。
0 =禁止，IROUT（P5.4）为普通的 I/O 引脚；
1 =使能，IROUT 输出引脚，且输出高电平。

Bit 0 **CREN**：IR 载波输出控制位。
0 =禁止，IROUT 输出高电平；
1 =使能，IROUT 输出 IR 载波信号。

✷ **注：IR 载波输出条件为 IREN=1 且 CREN=1，如果 CREN=1 而 IREN=0，IROUT（P5.4）引脚为普通 I/O 的引脚。**

9.2.2 IRC计数寄存器

8 位计数寄存器 IRC 控制 IR 的间隔时间。

0DBH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IRC	IRC7	IRC6	IRC5	IRC4	IRC3	IRC2	IRC1	IRC0
读/写	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W
复位后	0	0	0	0	0	0	0	0

✷ **注：在使能 IR 信号输出功能之前设置 IRC=IRR，以保证第一次循环是正确的。**

9.2.3 IRR自动装载寄存器

寄存器 IRR 控制**红外信号**的频率。IR 定时器具有自动装载功能，当 IRC 溢出，IRR 的值会自动装载到 IRC，为 IR 信号的周期提供了精确的计时标准。用户不需要在中断中复位 IRC。

IR 为双重缓存器设计，当程序设置一个新的 IRR 值时，其新值就保存在第一个缓存器中，直到 IR 溢出，其新值就移动到真正的 IRR 缓存器中。

0CDH	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IRR	IRR7	IRR6	IRR5	IRR4	IRR3	IRR2	IRR1	IRR0
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

IRR 初始值的计算如下：

IRR 初始值 = 256 –（IR 间隔时间 * 输入时钟）

✷ **注：输入时钟来自于外部晶振或内部 IHRC 8MHz。**

➢ **例：设置 IR 的频率是 38KHz，输入时钟 = 4MHz。**

IRR 初始值= 256 -（IR 间隔时间*输入时钟）

IR 间隔时间= 1/38KHz = 26.3us

输入时钟=外部晶振 4MHz

IRR = 256 - (26.3us * 4MHz)
= 150.8 ≈ 151

9.2.4 IRD IR占空比控制寄存器

寄存器 IRD 可以控制红外信号的占空比和低电平脉冲的宽度。当 IRC = IRD 时，红外信号由低电平脉冲变成高电平脉冲。当 IRC 溢出，高电平脉冲停止，低电平脉冲的宽度是 IRD ~ IRR，高电平脉冲的宽度是 256 ~ IRD。这样通过 IRR 和 IRD 寄存器就可以调整红外信号的频率和占空比（duty / cycle）。

0E8H	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
IRD	IRD7	IRD6	IRD5	IRD4	IRD3	IRD2	IRD1	IRD0
读/写	W	W	W	W	W	W	W	W
复位后	0	0	0	0	0	0	0	0

IRD 初始值的计算如下：

IRD 初始值 = IRR + (256-IRR) / (1/IR 占空比)

➢ **例：IR 的频率是 38KHz，占空比是 1/3，输入时钟 = 4MHz。**

IRD 初始值= IRR + (256-IRR) / (1/IR 占空比)

IRR（IR 的频率是 38KHz） = 151

IRD = 151 + (256-151)/(1/ (1/3))
= 186
= BAH

红外信号的基本列表。系统时钟 = 4MHz

IR 频率（KHz）	IRC IRR		IRD						频率错误的几率
			1/2 duty		1/3 duty		1/4duty		
	DEC	HEX	DEC	HEX	DEC	HEX	DEC	HEX	
32	131	83	193.50	C1	172.67	AC	162.25	A2	0.00%
36	145	91	200.50	C8	182.00	B6	172.75	AC	0.10%
38	151	97	203.50	CB	186.00	BA	177.25	B1	0.25%
39.2	154	9A	205.00	CD	188.00	BC	179.50	B3	0.04%
40	156	9C	206.00	CE	189.33	BD	181.00	B5	0.00%
56	185	B9	220.50	DC	208.67	D0	202.75	CA	0.60%

9.3 IR输出操作流程

- **设置 IRC 和 IRR，得到 IR 的周期。**

```
MOV     A, #IRCYCVAL
MOV     IRC, A
MOV     IRR, A
```

- **设置 IRD，得到 IR 的占空比。**

```
MOV     A, #IRDUTYVAL
MOV     IRD, A
```

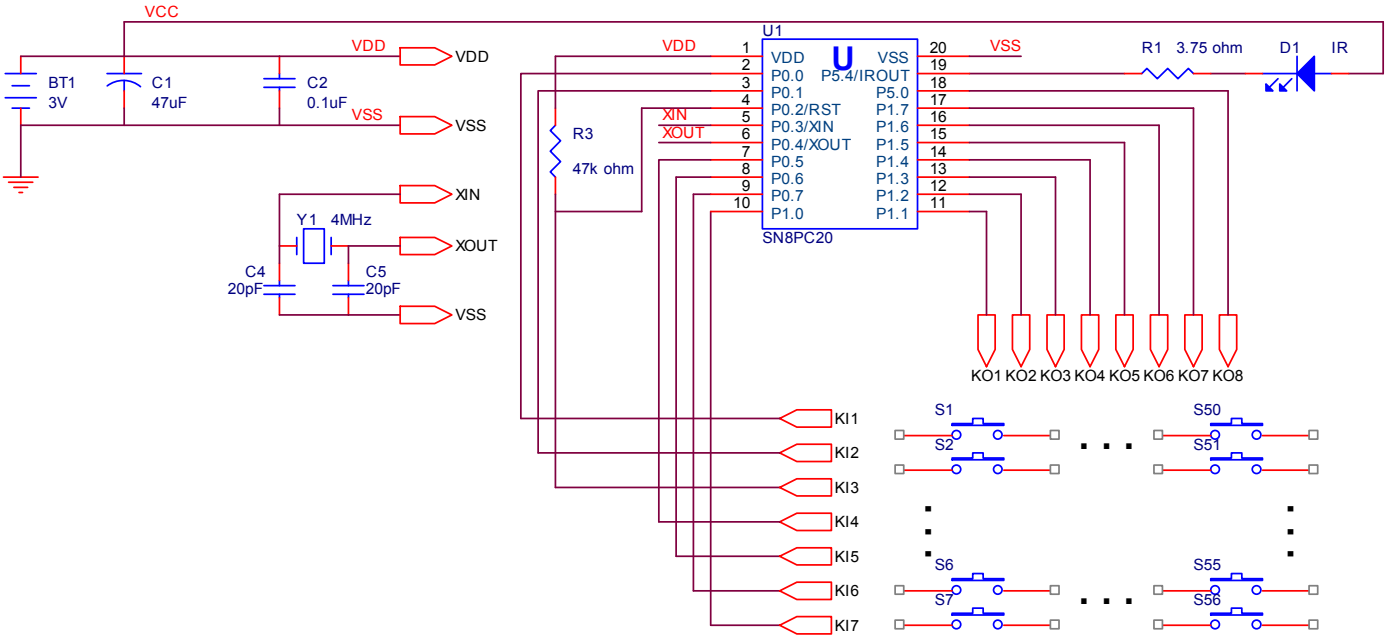
- **使能 IR 输出。**

```
BSET    FIREN       ;设置 IROUT（P5.4）引脚为 IR 功能输出引脚。
BSET    FCREN       ;设置 IR 输出载波信号。
```

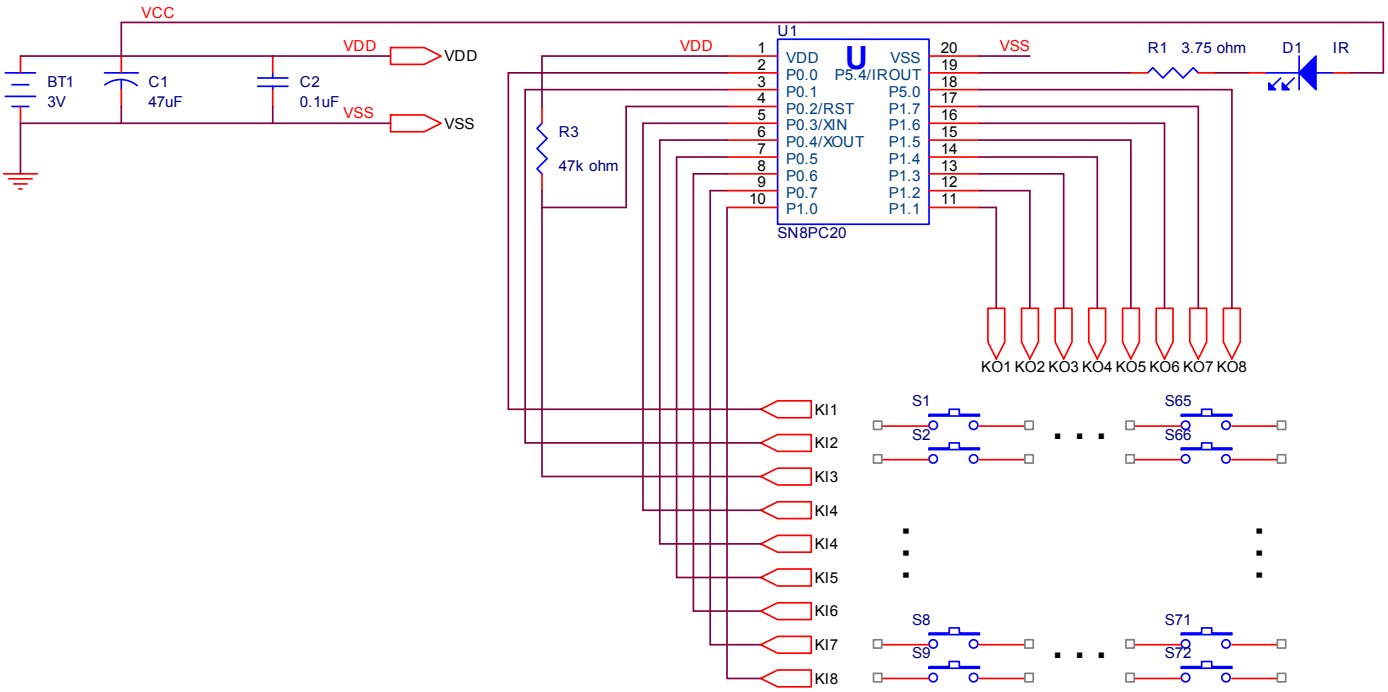
9.4 IR应用电路

Vss+1.5V 时，IROUT 引脚最大可以承受 400mA 灌电流。IR 驱动电路中的电阻为 3.75ohm（Vdd=3V）。电阻不能小于 3.75ohm，否则灌电流将超出规范。

晶振模式，最多 56 个按键：



IHRC_8M 模式，最多 72 个按键：



10 指令集

指令	指令格式		描述	C	DC	Z	周期
	MOV	A,M	A ← M。	-	-	√	1
M	MOV	M,A	M ← A。	-	-	-	1
O	B0MOV	A,M	A ← M (bank 0)。	-	-	√	1
V	B0MOV	M,A	M (bank 0) ← A。	-	-	-	1
E	MOV	A,I	A ← I	-	-	-	1
	B0MOV	M,I	M ← I。（M 仅适用于系统寄存器 R、Y、Z、RBANK、PFLAG。）	-	-	-	1
	XCH	A,M	A ←→M。	-	-	-	1+N
	B0XCH	A,M	A ←→M (bank 0)。	-	-	-	1+N
	MOVC		R, A ← ROM [Y,Z]。	-	-	-	2
	ADC	A,M	A ← A + M + C，如果产生进位则 C=1，否则 C=0。	√	√	√	1
A	ADC	M,A	M ← A + M + C，如果产生进位则 C=1，否则 C=0。	√	√	√	1+N
R	ADD	A,M	A ← A + M，如果产生进位则 C=1，否则 C=0。	√	√	√	1
I	ADD	M,A	M ← A + M，如果产生进位则 C=1，否则 C=0。	√	√	√	1+N
T	B0ADD	M,A	M (bank 0) ← M (bank 0) + A，如果产生进位则 C=1，否则 C=0。	√	√	√	1+N
H	ADD	A,I	A ← A + I，如果产生进位则 C=1，否则 C=0。	√	√	√	1
M	SBC	A,M	A ← A - M - /C，如果产生借位则 C=0，否则 C=1。	√	√	√	1
E	SBC	M,A	M ← A - M - /C，如果产生借位则 C=0，否则 C=1。	√	√	√	1+N
T	SUB	A,M	A ← A - M，如果产生借位则 C=0，否则 C=1。	√	√	√	1
I	SUB	M,A	M ← A - M，如果产生借位则 C=0，否则 C=1。	√	√	√	1+N
C	SUB	A,I	A ← A - I，如果产生借位则 C=0，否则 C=1。	√	√	√	1
	AND	A,M	A ← A 与 M。	-	-	√	1
L	AND	M,A	M ← A 与 M。	-	-	√	1+N
O	AND	A,I	A ← A 与 I。	-	-	√	1
G	OR	A,M	A ← A 或 M。	-	-	√	1
I	OR	M,A	M ← A 或 M。	-	-	√	1+N
C	OR	A,I	A ← A 或 I。	-	-	√	1
	XOR	A,M	A ← A 异或 M。	-	-	√	1
	XOR	M,A	M ← A 异或 M。	-	-	√	1+N
	XOR	A,I	A ← A 异或 I。	-	-	√	1
	SWAP	M	A (b3~b0, b7~b4) ←M(b7~b4, b3~b0)。	-	-	-	1
P	SWAPM	M	M(b3~b0, b7~b4) ← M(b7~b4, b3~b0)。	-	-	-	1+N
R	RRC	M	A ← M 带进位右移。	√	-	-	1
O	RRCM	M	M ← M 带进位右移。	√	-	-	1+N
C	RLC	M	A ← M 带进位左移。	√	-	-	1
E	RLCM	M	M ← M 带进位左移。	√	-	-	1+N
S	CLR	M	M ← 0。	-	-	-	1
S	BCLR	M.b	M.b ← 0。	-	-	-	1+N
	BSET	M.b	M.b ← 1	-	-	-	1+N
	B0BCLR	M.b	M(bank 0).b ← 0。	-	-	-	1+N
	B0BSET	M.b	M(bank 0).b ← 1。	-	-	-	1+N
	CMPRS	A,I	比较，如果相等则跳过下一条指令 C 与 ZF 标志位可能受影响。	√	-	√	1 + S
B	CMPRS	A,M	比较，如果相等则跳过下一条指令 C 与 ZF 标志位可能受影响。	√	-	√	1 + S
R	INCS	M	A ← M + 1，如果 A = 0，则跳过下一条指令。	-	-	-	1+ S
A	INCMS	M	M ← M + 1，如果 M = 0，则跳过下一条指令。	-	-	-	1+N+S
N	DECS	M	A ← M - 1，如果 A = 0，则跳过下一条指令。	-	-	-	1+ S
C	DECMS	M	M ← M - 1，如果 M = 0，则跳过下一条指令。	-	-	-	1+N+S
H	BTS0	M.b	如果 M.b = 0，则跳过下一条指令。	-	-	-	1 + S
	BTS1	M.b	如果 M.b = 1，则跳过下一条指令。	-	-	-	1 + S
	B0BTS0	M.b	如果 M(bank 0).b = 0，则跳过下一条指令。	-	-	-	1 + S
	B0BTS1	M.b	如果 M(bank 0).b = 1，则跳过下一条指令。	-	-	-	1 + S
	JMP	d	跳转指令，PC15/14 ← RomPages1/0，PC13~PC0 ← d。	-	-	-	2
	CALL	d	子程序调用指令，Stack ← PC15~PC0，PC15/14 ← RomPages1/0，PC13~PC0 ← d。	-	-	-	2
M	RET		子程序跳出指令，PC ← Stack。	-	-	-	2
I	RETI		中断处理程序跳出指令，PC ← Stack，使能全局中断控制位。	-	-	-	2
S	PUSH		进栈指令，保存 ACC 和工作寄存器。	-	-	-	1
C	POP		出栈指令，恢复 ACC 和工作寄存器。	√	√	√	1
	NOP		空指令，无特别意义。	-	-	-	1

注： **1. “M”是系统寄存器或 RAM，M 为系统寄存器时 N = 0，否则 N = 1。**
2.条件跳转指令的条件为真，则 S = 1，否则 S = 0。

11 电气特性

11.1 极限参数

Supply voltage (Vdd)..... - 0.3V ~ 6.0V
Input in voltage (Vin)..... Vss – 0.2V ~ Vdd + 0.2V
Operating ambient temperature (Topr)
SN8PC20P, SN8PC20S, SN8PC20X..... 0°C ~ + 70°C
Storage ambient temperature (Tstor) ..... –40°C ~ + 125°C

11.2 电气特性

DC CHARACTERISTIC

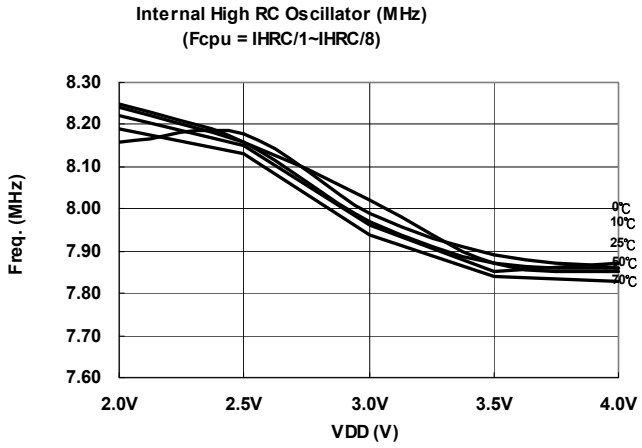
(All of voltages refer to Vss, Vdd = 3.0V, fosc = 4MHz,fcpu=1MHZ,ambient temperature is 25°C unless otherwise note.)

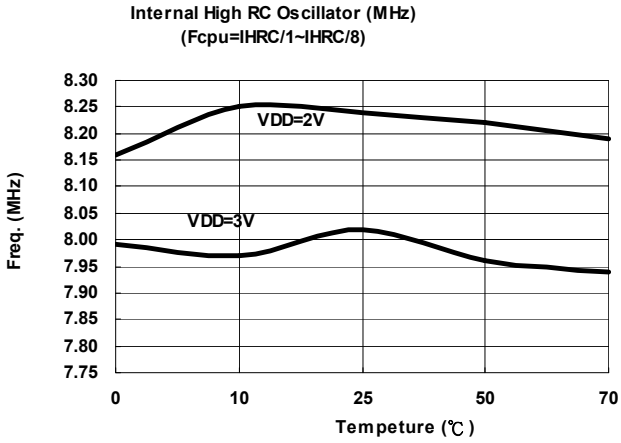
PARAMETER	SYM.	DESCRIPTION		MIN.	TYP.	MAX.	UNIT
Operating voltage	Vdd	Normal mode, Vpp = Vdd, 25°C, Fcpu = 2mips.		1.8	3.0	3.6	V
RAM Data Retention voltage	Vdr			1.5	-	-	V
Vdd rise rate	Vpor	Vdd rise rate to ensure internal power-on reset		0.05	-	-	V/ms
Input Low Voltage	ViL1	All input ports		Vss	-	0.3Vdd	V
	ViL2	Reset pin		Vss	-	0.2Vdd	V
Input High Voltage	ViH1	All input ports		0.7Vdd	-	Vdd	V
	ViH2	Reset pin		0.9Vdd	-	Vdd	V
Reset pin leakage current	Ilekg	Vin = Vdd		-	-	2	uA
I/O port pull-up resistor	Rup	Vin = Vss , Vdd = 3V		100	200	300	KΩ
I/O port input leakage current	Ilekg	Pull-up resistor disable, Vin = Vdd		-	-	2	uA
I/O output source current sink current	IoH	Vop = Vdd – 0.5V		8	10	-	mA
	IoL1	Vop = Vss + 0.5V		8	12	-	
	IoL2	Vop = Vss + 1.5V, IR output pin		300	400	-	mA
INTn trigger pulse width	Tint0	INT0 interrupt request pulse width		2/fcpu	-	-	cycle
Supply Current	Idd1	Run Mode (No loading, Fcpu = Fosc/4)	Vdd= 3V, 4Mhz	-	1	2	mA
	Idd2	Slow Mode (Internal low RC, Stop high clock)	Vdd= 3V, 10Khz	-	5	10	uA
	Idd3	Sleep Mode	Vdd= 3V, 25°C	-	1	2	uA
	Idd4	Green Mode (No loading, Fcpu = Fosc/4 Watchdog Disable)	Vdd= 3V, 4Mhz	-	0.25	0.5	mA
			Vdd=3V, ILRC 10Khz ,	-	3	6	uA
Internal High Oscillator Freq.	Fihrc	Internal Hihg RC (IHRC)	25°C, Vdd= 3V, Fcpu = 1MHz	7.84	8	8.16	Mhz
LVD Voltage	Vdet0	Low voltage reset level.		-	-	1.8	V

" *" These parameters are for design reference, not tested.

11.3 特性曲线图

本章所列的各曲线图仅作设计参考，其中给出的部分数据可能超出了芯片指定的工作范围，为保证芯片的正常工作，请严格参照电气特性说明。





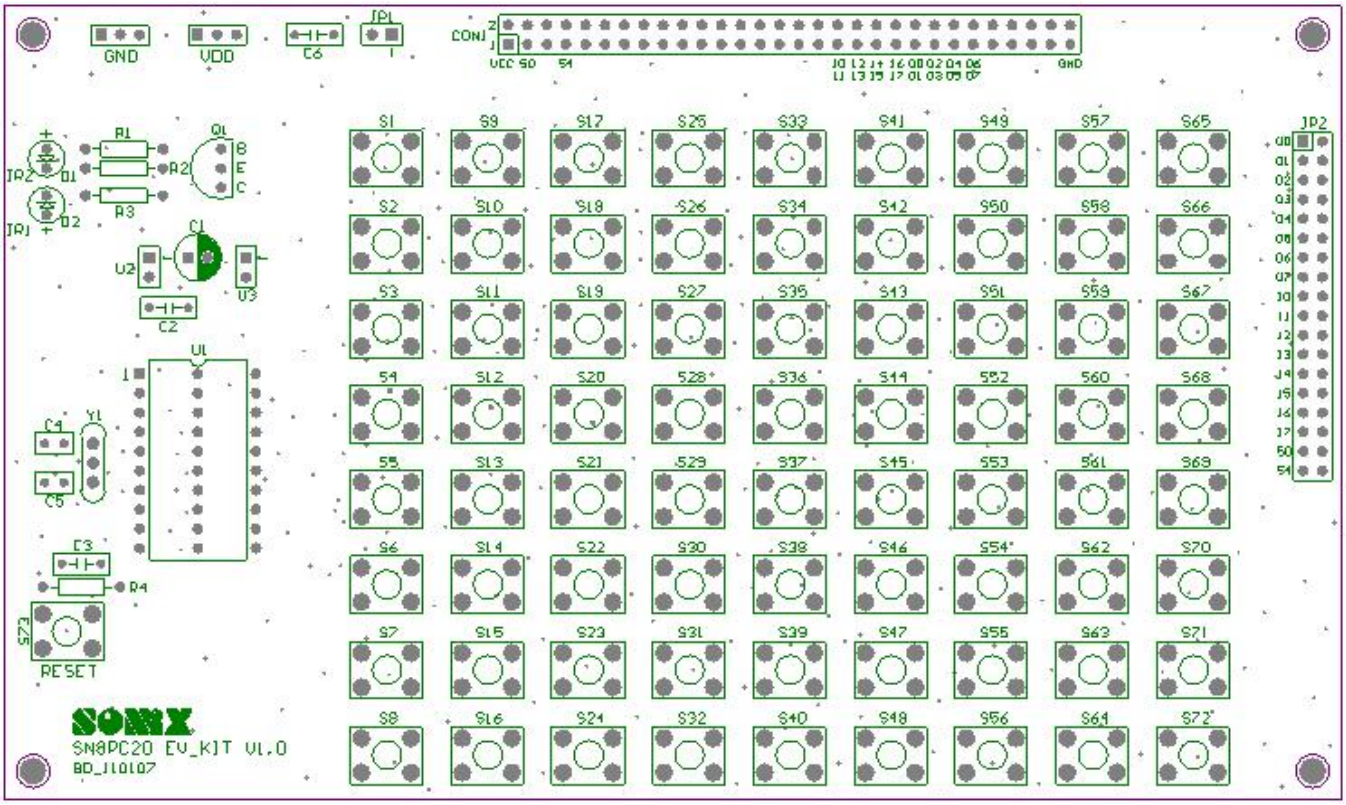
12 开发工具

SONIX 为 SN8PC20 的开发提供了在线仿真器（ICE）、集成开发环境（IDE）和 EV-Kit。ICE 和 EV-Kit 是外部硬件装置，IDE 有一个友好的界面，为用户调试提供方便。各开发工具的版本如下：

- **ICE：SN8ICE2K；**
- **EV-kit：SN8PC20 EV-kit V1.0；**
- **IDE：SONiX IDE M2IDE_V115 或更晚的版本；**
- **Writer：MPIII WRITER-LV。**

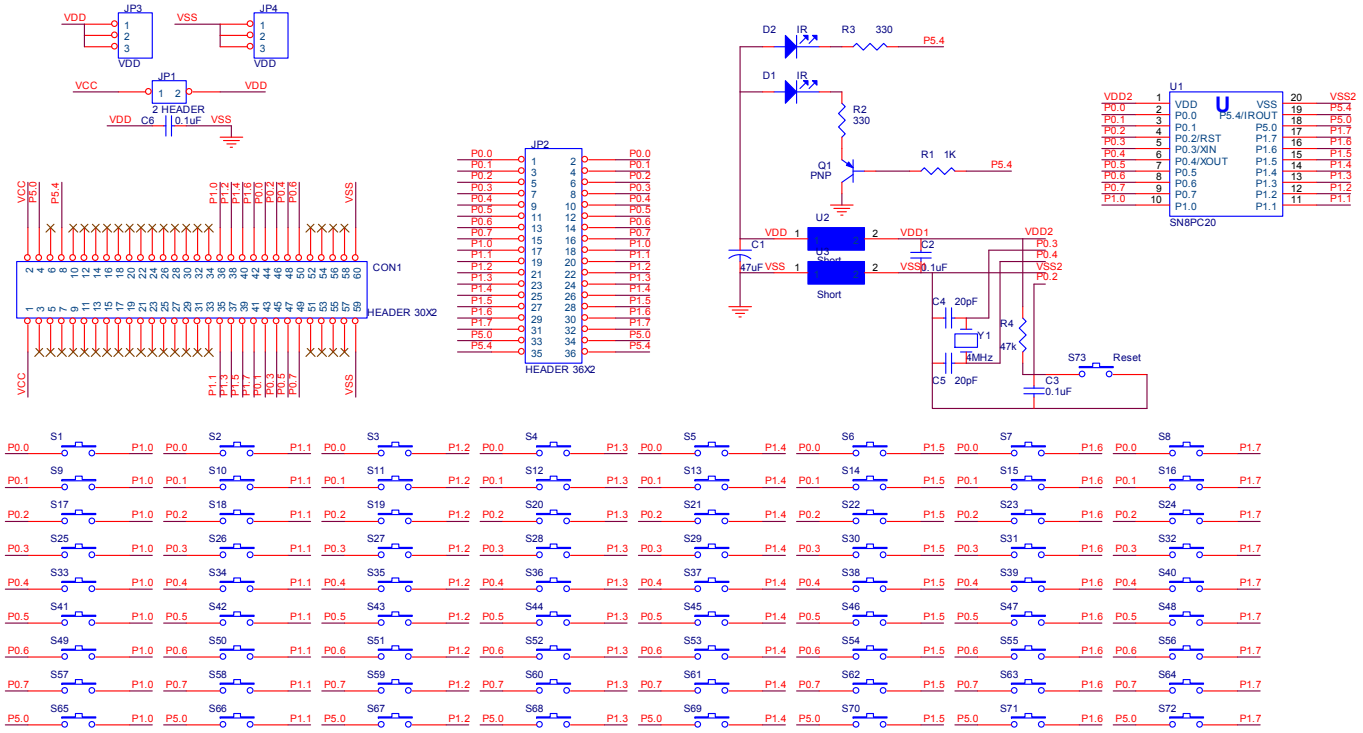
12.1 SN8PC20 EV-KIT

SN8PC20 EV-kit 包括 ICE 接口、GPIO 接口和 IR 驱动模块。SN8PC20 EV-Kit 的 PCB 图如下所示：



- CON1：连接到 SN8ICE2K。
- JP1：EV-Kit 电源输入端。通过 JP1 把 VCC 和 VDD 短接到一处。VCC 的电源来自 SN8ICE2K，VDD 则为 EV-Kit 的电源。
- U1：SN8PC20 DIP 和 SSOP 封装形式的芯片插座，连接到用户目标板。
- U2/U3：内部短路跳线。仿真时可以忽略。
- S1~S72：遥控按键。
- D1/R1/R2/Q1：ICE 仿真时的 IR 驱动电路。
- D2/R3：SN8PC20 实际芯片测试时的 IR 驱动电路。这两个元器件不是 SN8PC20 EV-Kit 上的器件。

SN8PC20 EV-kit 原理图：



12.2 ICE和EV-KIT应用注意事项

SN8PC20 EV-kit 包括矩阵按键电路和 IR 驱动电路。

- IR 驱动电路模块包括两种：一种是使用 ICE 仿真时驱动 IR 的载波信号；另外一种是使用 SN8PC20 实际芯片测试时，和 ICE 断开。该 EV-Kit 类似于一个真正的遥控器。R3 的阻值最好大于 3.75Ω，以免超出 400mA 的灌电流（@Vdd = 3V）。
- 若使用 SN8PC20 实际芯片测试时，采用的是 455KHz 的振荡模式，则 C4（XIN）最好为 47pF，C5（XOUT）最好为 20pF。

13 OTP烧录信息

13.1 烧录转接板信息

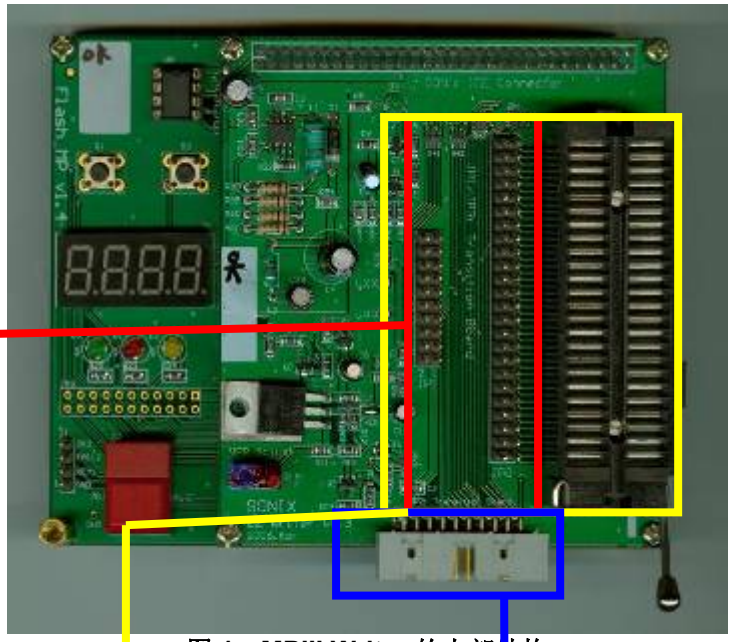


图 1　MPIII Writer 的内部结构

Writer 上板 JP1/JP3

Writer 上板 JP1/JP3

Writer 上板 JP2

注 1：JP1 连接 MP 烧录转接板，JP3 连接 OTP MCU。

注 2：JP2 连接外部烧录转接板。当 OTP MCU 的 PIN 超过 48PIN，或者烧录 Dice MCU 时，请采用外部烧录转接板，连接到 JP2 进行烧录。

下面两个图演示了如何焊接烧录转接板。

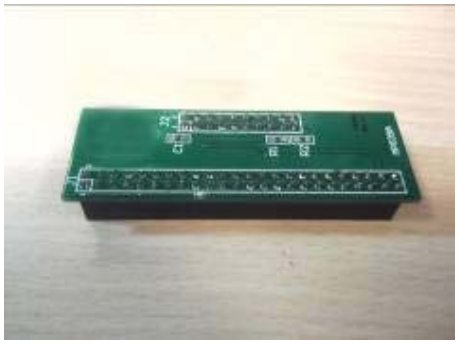


图 2

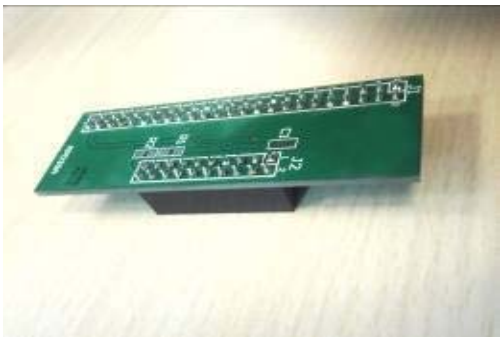


图 3

注：
1、印有 IC 型号的这一面为转接板的正面。
2、180 度的母座必须焊接在 MP 转接板的背面。请参考图 2 和图 3。

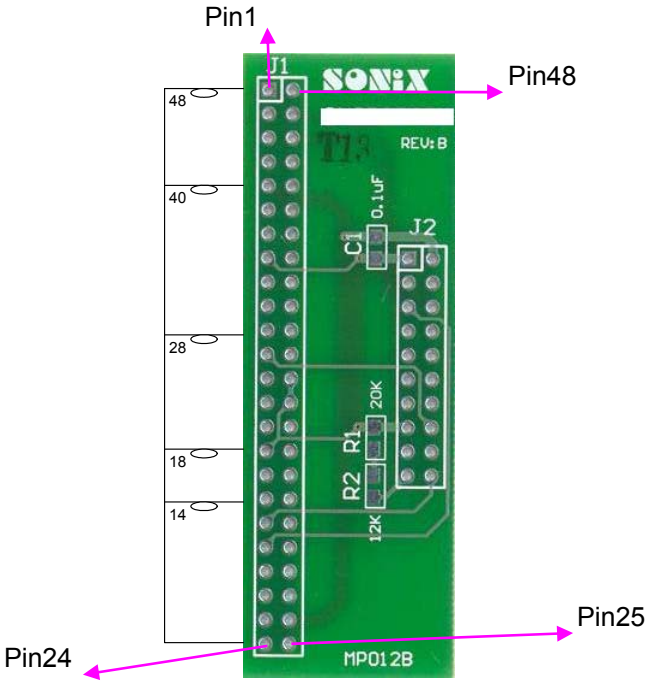


图 4　MP 转接板（连接到 JP1&JP3）

JP3（连接 48-pin text tool）

DIP 1	1	48	DIP48
DIP 2	2	47	DIP47
DIP 3	3	46	DIP46
DIP 4	4	45	DIP45
DIP 5	5	44	DIP44
DIP 6	6	43	DIP43
DIP 7	7	42	DIP42
DIP 8	8	41	DIP41
DIP 9	9	40	DIP40
DIP10	10	39	DIP39
DIP11	11	38	DIP38
DIP12	12	37	DIP37
DIP13	13	36	DIP36
DIP14	14	35	DIP35
DIP15	15	34	DIP34
DIP16	16	33	DIP33
DIP17	17	32	DIP32
DIP18	18	31	DIP31
DIP19	19	30	DIP30
DIP20	20	29	DIP29
DIP21	21	28	DIP28
DIP22	22	27	DIP27
DIP23	23	26	DIP26
DIP24	24	25	DIP25

JP1/JP2

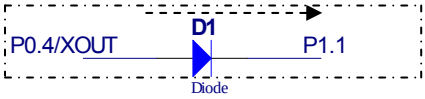
VDD	1	2	VSS
CLK/PGCLK	3	4	CE
PGM/OTPCLK	5	6	OE/ShiftDat
D1	7	8	D0
D3	9	10	D2
D5	11	12	D4
D7	13	14	D6
VDD	15	16	VPP
HLS	17	18	RST
-	19	20	ALSB/PDB

JP1 连接 MP 烧录转接板
JP2 连接外部转接板

13.2 烧录引脚信息

SN8PC20 的烧录信息				
单片机名称		SN8PC20P/S/X		
MPIII Writer		OTP IC / JP3 引脚配置		
JP1/JP2 Pin Number	JP1/JP2 Pin Name	IC Pin Number	IC Pin Name	JP3 Pin Number
1	VDD	1	VDD	15
2	GND	20	VSS	34
3	CLK	12	P1.2	26
4	CE	-	-	-
5	PGM	10	P1.0	24
6	OE	13	P1.3	27
7	D1	-	-	-
8	D0	-	-	-
9	D3	-	-	-
10	D2	-	-	-
11	D5	-	-	-
12	D4	-	-	-
13	D7	-	-	-
14	D6	-	-	-
15	VDD	-	-	15
16	VPP	4	RST	18
17	HLS	-	-	-
18	RST	-	-	-
19	-	-	-	-
20	ALSB/PDB	6, 11	P0.4/P1.1	20,25

✷ **注：在烧录转接板的 PIN6（P04/XOUT）和 PIN11（P1.1）之间必须连接一个二极管，二极管 D1 的连接方向如下图所示：**

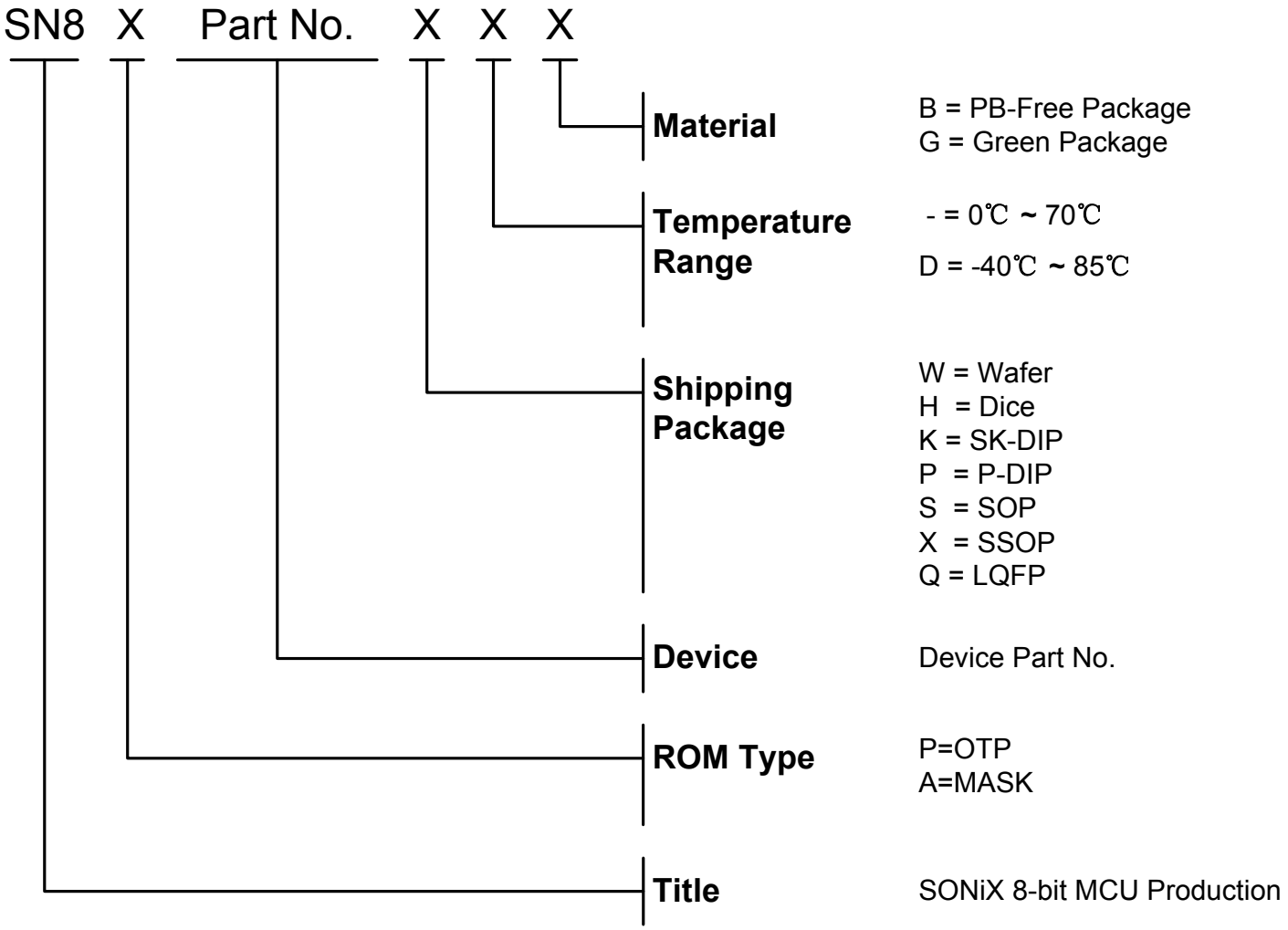


14 单片机正印命名规则

14.1 概述

SONiX 8 位单片产品具有多种型号，本章将给出所有 8 位单片机分类命名规则，适用于空片 OTP 型单片。

14.2 单片机型号说明

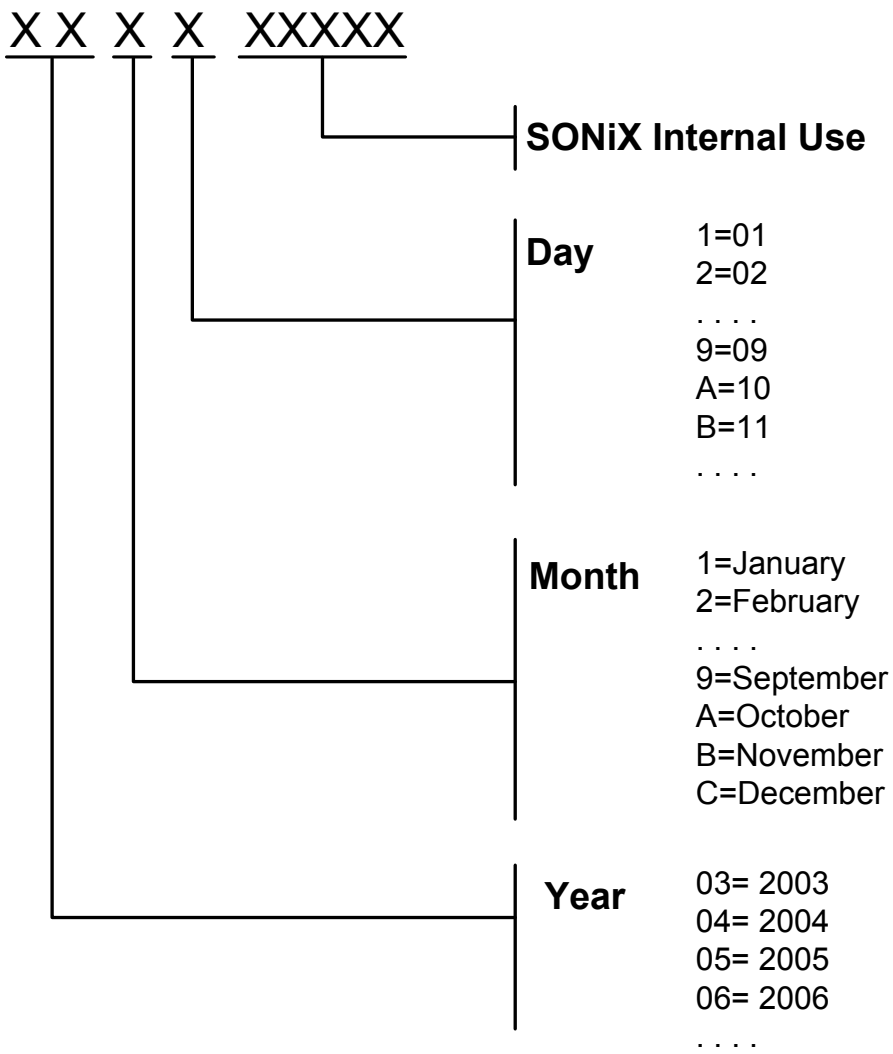


✷ 注：SN8PC20 不支持-40ºC~85ºC 的温度范围。

14.3 命名举例

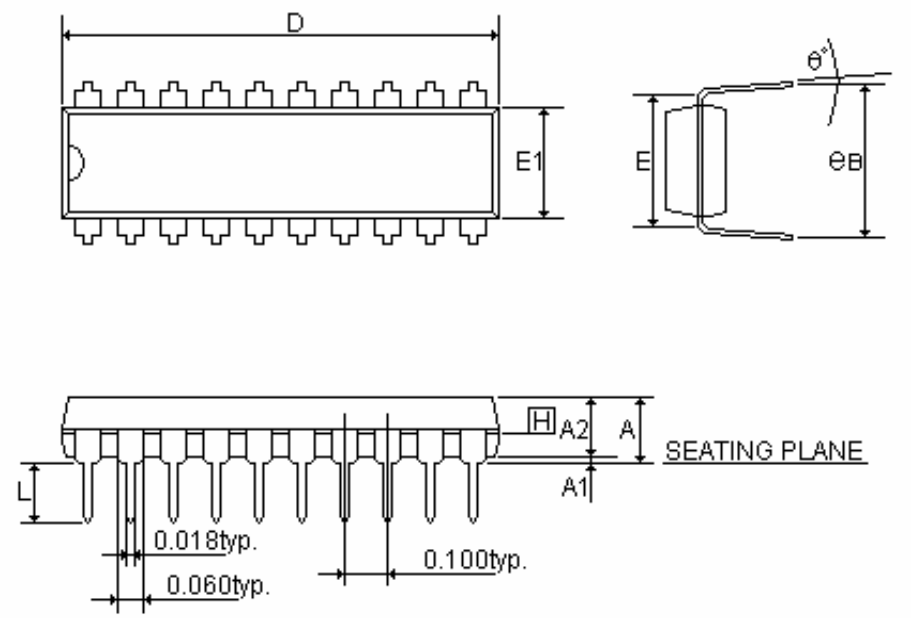
名称	ROM 类型	单片机型号（Device）	封装形式	温度范围	封装材料
SN8PC20PG	OTP	C20	P-DIP	0℃~70℃	绿色封装（Green Package）
SN8PC20SG	OTP	C20	SOP	0℃~70℃	绿色封装（Green Package）
SN8PC20XG	OTP	C20	SSOP	0℃~70℃	绿色封装（Green Package）
SN8PC20W	OTP	C20	Wafer	0℃~70℃	-
SN8PC20H	OTP	C20	Dice	0℃~70℃	-

14.4 日期码规则



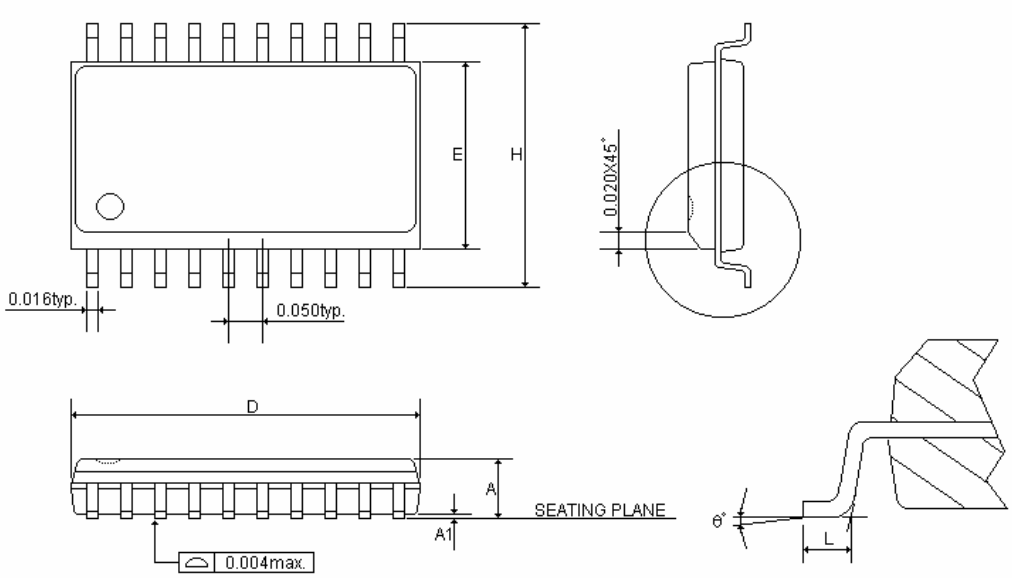
15 封裝信息

15.1 P-DIP 20 PIN



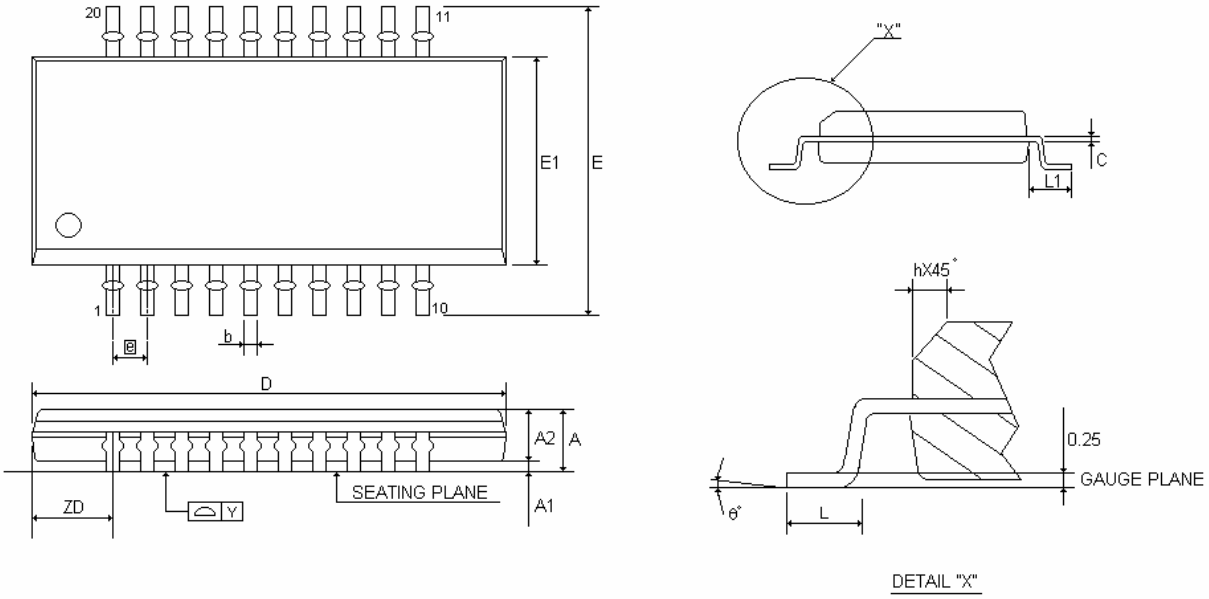
SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	-	-	0.210	-	-	5.334
A1	0.015	-	-	0.381	-	-
A2	0.125	0.130	0.135	3.175	3.302	3.429
D	0.980	1.030	1.060	24.892	26.162	26.924
E		0.300			7.620	
E1	0.245	0.250	0.255	6.223	6.350	6.477
L	0.115	0.130	0.150	2.921	3.302	3.810
e B	0.335	0.355	0.375	8.509	9.017	9.525
θ°	0°	7°	15°	0°	7°	15°

15.2 SOP 20 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.093	0.099	0.104	2.362	2.502	2.642
A1	0.004	0.008	0.012	0.102	0.203	0.305
D	0.496	0.502	0.508	12.598	12.751	12.903
E	0.291	0.295	0.299	7.391	7.493	7.595
H	0.394	0.407	0.419	10.008	10.325	10.643
L	0.016	0.033	0.050	0.406	0.838	1.270
θ°	0°	4°	8°	0°	4°	8°

15.3 SSOP 20 PIN



SYMBOLS	MIN	NOR	MAX	MIN	NOR	MAX
	(inch)			(mm)		
A	0.053	0.063	0.069	1.350	1.600	1.750
A1	0.004	0.006	0.010	0.100	0.150	0.250
A2	-	-	0.059	-	-	1.500
b	0.008	0.010	0.012	0.200	0.254	0.300
c	0.007	0.008	0.010	0.180	0.203	0.250
D	0.337	0.341	0.344	8.560	8.660	8.740
E	0.228	0.236	0.244	5.800	6.000	6.200
E1	0.150	0.154	0.157	3.800	3.900	4.000
[e]		0.025			0.635	
h	0.010	0.017	0.020	0.250	0.420	0.500
L	0.016	0.025	0.050	0.400	0.635	1.270
L1	0.039	0.041	0.043	1.000	1.050	1.100
ZD		0.059			1.500	
Y	-	-	0.004	-	-	0.100
θ°	0°	-	8°	0°	-	8°

SONiX 公司保留对以下所有产品在可靠性，功能和设计方面的改进作进一步说明的权利。SONiX 不承担由本手册所涉及的产品或电路的运用和使用所引起的任何责任，SONiX 的产品不是专门设计来应用于外科植入、生命维持和任何 SONiX 产品的故障会对个体造成伤害甚至死亡的领域。如果将 SONiX 的产品应用于上述领域，即使这些是由 SONiX 在产品设计和制造上的疏忽引起的，用户应赔偿所有费用、损失、合理的人身伤害或死亡所直接或间接产生的律师费用，并且用户保证 SONiX 及其雇员、子公司、分支机构和销售商与上述事宜无关。

总公司：
地址：台湾新竹县竹北市台元街 36 号 10 楼之一
电话：886-3-5600-888
传真：886-3-5600-889
台北办事处：
地址：台北市松德路 171 号 15 楼之 2
电话：886-2-2759 1980
传真：886-2-2759 8180
香港办事处：
地址：香港新界沙田沙田乡宁会路 138＃新城市中央广场第一座 7 楼 705 室
电话：852-2723 8086
传真：852-2723 9179
松翰科技（深圳）有限公司
地址：深圳市南山区高新技术产业园南区 T2-B 栋 2 层
电话：86-755-2671 9666
传真：86-755-2671 9786
技术支持：
Sn8fae@SONiX.com.tw